Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

Robotic mobile manipulation in a completely unknown environment

Relatore: Prof. Paolo Rocco
Correlatore: Ing. Giovanni Massimo Buizza Avanzini

Tesi di Laurea Magistrale di:
Roberto Ancona, matricola 798599

Anno Accademico 2013-2014

To my family

Contents

List of Figures

Acknowledgements

I wish to thank my advisor, prof. Paolo Rocco, for the professionalism and the precious guidance he demonstrated during the development of my thesis. Professor Rocco gave me the opportunity of working for a long time in the ***ME****chatronics and* ***R****obotics* ***L****aboratory for* ***In****novation,* an experience that I think will be very valuable in my future professional career.

A special and sincere thanks goes to Giovanni Buizza Avanzini, for reviewing my work and for all the interesting ideas and thoughts, regarding mobile manipulation and robotics in general, he shared with me. His precise support was always available, even when he was many kilometres away from Milan.

Thanks to my colleagues Diego, Lorenzo and all the other guys I have met at the automation laboratory of the Politecnico di Milano, working with them has been truly funny and pleasant.

Thanks to Paolo and Vespe, the third musketeer is deeply grateful to them for always sticking around.

Thanks to Roberto, Stefano and Francesca, my lifetime friends.

I own much of what I am to my family, no words can express the gratitude I feel for their unconditional love, support and understanding.

Finally, I wish to thank Michele, because he, more then anyone, knows everything about *il mio robottino*.

Abstract

Thanks to the high number of degrees of freedom and to the ability to freely navigate in the environment, mobile manipulators exhibit a virtually unlimited reachable workspace, while also presenting a remarkable level of dexterity, namely the capability of performing manipulation tasks. These features make mobile manipulators particularly suited to carry out autonomous task in an unknown and dynamic environment.

Within this scenario, a framework to execute autonomous pick and place operations in a completely unknown environment has been developed in this work, exploiting a KUKA youBot mobile manipulator as experimental platform. Both the environment and information such as shape, position and dimension of the objects to be manipulated are a-priori unknown to the robot. All the necessary data are then acquired on-line through a depth camera and several proximity sensors. A grasp synthesis process has been introduced to compute optimal grasping configurations on the fly, based on data acquired by an Xtion Pro Live depth sensor. Optimization of the whole robot manipulability, with the purpose of increasing the coordination between base and arm, and minimization of the mobile platform motions are obtained by exploiting the redundancy of the KUKA youBot mobile manipulator. A technique based on rotational potential fields has also been adopted to allow the robot to navigate autonomously without colliding with the environment. Both simulated and experimental validations of the proposed approach demonstrate the effectiveness and reliability of our implementation in dealing with manipulation tasks in unknown environments.

Estratto in lingua italiana

Al giorno d'oggi viene fatto ampio uso di sistemi robotici in ambito industriale per eseguire operazioni pericolose, ripetitive e pesanti. L'introduzione di robot nel contesto manifatturiero, infatti, consente di migliorare la qualità dei prodotti, le condizioni di lavoro, e porta ad un uso ottimizzato delle risorse. Le linee di produzione delle attuali industrie, tuttavia, sono dotate di manipolatori fissi e dedicati, che risultano poco flessibili, poiché ripetono meccanicamente le stesse operazioni per tutta la durata del processo di produzione. Recentemente, l'inadeguatezza di questi manipolatori sta diventando sempre più evidente alla luce della globalizzazione dei mercati e della sempre crescente domanda di diversificazione di prodotto, che richiede uno spostamento dal paradigma di produzione di massa al paradigma di produzione personalizzata. Le correnti tecnologie dell'automazione rendono difficile e costoso scalare la produzione e la varietà dei prodotti in base alla volatilità del mercato. Inoltre, a causa di questa ridotta flessibilità, il mercato fatica a trovare concrete soluzioni commerciali nell'ambito della robotica di servizio, nel cui contesto i robot devono assistere gli esseri umani nell'ambiente domestico e lavorativo.
Per superare queste problematiche, le future tecnologie dovranno consentire ai sistemi robotici di operare in ambienti sconosciuti e dinamici, cooperare con gli esseri umani e, più in generale, costituire dei sistemi *general purpose* in grado di eseguire una gran varietà di compiti e funzionalità. La manipolazione mobile costituisce una valida alternativa per raggiungere questi obbiettivi sia nella robotica industriale, sia nella robotica di servizio. I manipolatori mobili sono robot costituiti da uno o più bracci robotici montati su una piattaforma mobile. La composizione di capacità locomotive e manipolative assicura a questi robot un maggior livello di destrezza e uno spazio di lavoro raggiungibile virtualmente illimitato. Per queste ragioni, i manipolatori mobili, confrontati coi tradizionali manipolatori fissi, risultano più adatti a svolgere una ampia varietà di compiti nei più svariati ambienti.
La navigazione è una attività fondamentale per i robot mobili, che devono essere in grado di muoversi in modo sicuro da una posizione all'altra dell'ambiente, evitando possibili collisioni con ostacoli presenti in esso. In un contesto industriale, i manipolatori mobili devono spostarsi autonomamente tra le diverse stazioni di lavoro che formano la linea di produzione, evitando collisioni con altri robot fissi o mobili, macchine e operatori umani. I robot,

quindi, devono essere provvisti di una strategia di navigazione, che, basandosi sulle percezioni sensoriali acquisite in tempo reale, permetta loro di muoversi all'interno di ambienti dinamici e sconosciuti.
I manipolatori mobili, inoltre, devono interagire con gli oggetti presenti nello spazio di lavoro per eseguire operazioni di manipolazione e assemblaggio. Attualmente, nel campo della robotica industriale, ogni singola operazione viene accuratamente programmata da operatori umani e poi viene eseguita dai robot, che meccanicamente la ripetono. Un procedimento più adatto ad eseguire operazioni di manipolazione in modo autonomo sarebbe specificare solamente lo stato finale desiderato degli oggetti, e lasciare al robot il compito di pianificare la corretta sequenza di operazioni necessaria per completare l'incarico assegnato. Per sviluppare questo tipo di approccio, i robot devono essere in grado di interagire autonomamente con una grande varietà di oggetti, decidendo il modo migliore per afferrarli e riposizionarli in una nuova allocazione stabile. La tematica del *grasping*, ovvero come il robot debba afferrare un certo oggetto, è particolarmente critica se eseguita in modo autonomo. Infatti il robot, basandosi su un modello virtuale degli oggetti, deve pianificare e poi eseguire delle pose di grasping che permettano una presa sicura e stabile, nonostante queste cambino notevolmente da oggetto a oggetto.
Abilità di navigazione e manipolazione simili sono necessarie nel contesto della robotica di servizio, dove ai manipolatori mobili è richiesto di assistere esseri umani, non necessariamente esperti nel campo ICT, all'interno di ambienti e situazioni di tutti i giorni. Rispetto agli spazi di lavoro industriali, la navigazione autonoma di robot all'interno di case o uffici è ancora più complessa, in quanto essi costituiscono ambienti caotici, pieni di ostacoli e caratterizzati da passaggi stretti. Anche la manipolazione di oggetti risulta più critica nell'ambito della robotica di servizio, dove il robot deve essere in grado di eseguire svariate operazioni interagendo con oggetti completamente sconosciuti. Infatti, a differenza dalle situazioni industriali, in cui un modello completo ed accurato degli oggetti da manipolare è dato a priori, nella robotica di servizio questi modelli devono essere ricostruiti in tempo reale dal sottosistema sensoriale del robot, costituito da telecamere e sensori di profondità. Strategie euristiche devono poi essere usate per pianificare e controllare l'operazione di manipolazione, basandosi sul modello grezzo ricostruito in precedenza.
Una ulteriore tematica caratteristica riguardante la manipolazione mobile consiste nella risoluzione della ridondanza cinematica. Una ridondanza cinematica si manifesta quando il numero totale di gradi di libertà di un sistema meccanico supera quello strettamente necessario per eseguire una

certa operazione. Poiché i manipolatori mobili aggiungono ai gradi di libertà del braccio robotico i gradi di libertà della base mobile, di norma costituiscono dei robot ridondanti. La ridondanza permette al robot di eseguire la stessa operazione in infiniti modi diversi: scegliere tra questi una modalità che ottimizzi un voluto comportamento, è un problema importante da considerare per sfruttare al meglio le notevoli possibilità offerte dalla manipolazione mobile. Opportune tecniche di ridondanza possono essere utilizzate, ad esempio, per mantenere i valori di giunto all'interno dei loro vincoli fisici, mantenere il robot in una configurazione dove possiede una buona capacità di eseguire compiti di manipolazione, evitare collisioni con ostacoli o minimizzare il consumo di energia.

L'obiettivo principale di questa tesi consiste nello studiare come un robot mobile, dotato di una braccio robotico, possa eseguire compiti di manipolazione in un ambiente completamente sconosciuto. Si è prefissato di analizzare come la ridondanza, caratteristica comune dei manipolatori mobili, possa essere usata per migliorare la coordinazione tra braccio robotico e piattaforma mobile, e come possa essere sfruttata per supportare e ottimizzare le operazioni di manipolazione in generale. E' stato individuato, inoltre, il requisito di sviluppare una tecnica di navigazione che consenta al robot di muoversi in sicurezza evitando collisioni con l'ambiente circostante. Infine, è stato definito l'obiettivo di studiare come un manipolatore mobile possa essere adoperato, nel contesto della robotica industriale e di servizio, per afferrare oggetti con forma e allocazione non nota a priori.

In questo lavoro è stato sviluppato un sistema di manipolazione mobile che permette di eseguire in modo autonomo operazioni di pick-and-place utilizzando la piattaforma di ricerca costituita dal robot *KUKA youBot*. KUKA youBot è un robot composto da una base mobile omnidirezionale e da un manipolatore seriale a cinque assi, per un totale di otto gradi di libertà complessivi. La descrizione della ridondanza del robot è stata introdotta a livello cinematico, tramite la definizione di un insieme di parametri di ridondanza. Questi parametri, che descrivono il significato fisico dei gradi libertà ridondanti, possono essere arbitrariamente assegnati lasciando invariata la configurazione dell'end-effector del robot. La ridondanza del robot è stata utilizzata per ottimizzare la manipolabilità dell'intero manipolatore mobile, con l'intento di migliorare il coordinamento tra base e braccio. Inoltre, la ridondanza è stata utilizzata per minimizzare i movimenti della base mobile, mantenere i giunti all'interno dei loro limiti fisici e per tenere un certo obiettivo all'interno del campo visivo del sensore video

utilizzato. Per conferire al robot la capacità di navigare autonomamente, è stata sviluppata una strategia di controllo reattiva basandosi sul concetto di campi virtuali di forza rotazionali, che consentono al robot di aggirare gli ostacoli percepiti, invece che semplicemente essere respinto da essi. In questo modo è stato possibile superare il problema di minimo locale, caratteristico delle strategie di navigazione basate su campi di forza virtuali. Un insieme di sensori di prossimità montati attorno alla base mobile del robot, o un singolo dispositivo video di profondità sono stati usati indifferentemente per acquisire i dati necessari alla navigazione. Infine, è stato realizzato un robusto ed efficiente processo di sintesi di pose di grasping con l'intento di eseguire operazioni di manipolazione in un ambiente completamente sconosciuto, dove le informazioni riguardanti forma, dimensione, posizione e orientamento degli oggetti da afferrare non sono fornite in anticipo al robot. Il processo di sintesi del grasping sviluppato, utilizzando il sensore di profondità ASUS Xtion Live Pro, ricostruisce un modello virtuale degli oggetti presenti nell'ambiente e procede poi alla generazione di configurazioni di grasping ammissibili. Criteri euristici sono stati adottati per valutare la stabilità delle varie configurazioni, in modo da selezionare ed eseguire quelle che dimostrano maggior garanzia di successo. Grazie al framework realizzato, il robot KUKA youBot è in grado di esplorare autonomamente l'ambiente, riscostruire il modello virtuale degli oggetti da manipolare, pianificare ed eseguire correttamente le operazioni di pick-and-place.

La struttura della tesi è organizzata come segue: il Capitolo 1 introduce tematiche e motivazioni legate alla manipolazione mobile, nel Capitolo 2 viene introdotta la cinematica del robot KUKA youBot; possibili strategie di risoluzione della ridondanza cinematica vengono proposte nel Capitolo 3; la strategia di navigazione basata su campi di forza virtuale è presentata nel Capitolo 4; il processo sviluppato di sintesi del grasping viene illustrato nel Capitolo 5; nel Capitolo 6 vengono proposti alcuni dettagli implementativi riguardanti il sistema autonomo di pick-and-place realizzato; nel Capitolo 7 vengono presentati i risultati sperimentali ottenuti.

Chapter 1

An introduction to mobile manipulation

"Why do plants not have brains?
The answer is actually quite simple: they don't have to move."

Lewis Wolpert

1.1 Overview

Nowadays robots are widely used in industry to perform dangerous, dull and heavy tasks. Indeed, robot-based manufacturing increases product quality, improves work conditions, and leads to an optimized use of resources. However, the common production lines of today industries are equipped with fixed and dedicated manipulators, which result rather inflexible, as they repeat continuously the same task during all the production lifetime. In recent years, the inflexibility and inadequacy of industrial robotics has become more and more evident due to globalization of markets, trade instability, e-commerce and explosion of product variety, which leads to a shift in paradigm from mass production to customized production. The current automation practices, make it difficult, time consuming, and costly to change the type of products manufactured and to scale the production up and down in response to market volatility [1].

Moreover, the lack of flexibility causes the robotics market to struggle in finding concrete commercial solutions in the area of service robotics, where robotic systems should be devoted to assist humans in home-care and health-care fields. As shown in Figure 1.1, the future industrial robotics should maintain its efficiency to guarantee big production volumes, while enjoying a larger degree of flexibility to deal with product variety. Service robotics, instead, can afford to have less efficient performance, but must be able to execute a great variety of tasks and operations.

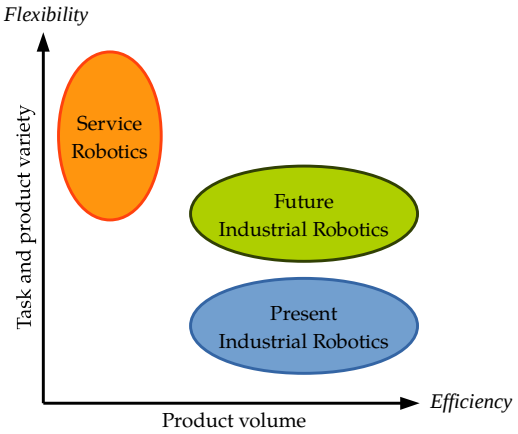


Figure 1.1 **Efficiency and flexibility relation in industrial and service robotics.**

In order to achieve these objectives, the future robotics technologies should autonomously operate in unknown and dynamic environments, cooperate with humans and be part of general purposes systems, capable of executing great variety of tasks. *Mobile manipulation* has been indicated [2] [3] [4] [5] as a convincing concept to achieve these objectives both in industrial and service robotics. Mobile manipulators are composed of one or more robotic arms mounted on a mobile platform. Indeed, the composition of locomotion a manipulation abilities ensures an increased level of dexterity and a virtually unlimited reachable workspace to mobile manipulators. Compared to traditional industrial robots, mobile manipulators are thus more suitable to adapt to changing environments and perform a wide variety of manufacturing tasks.

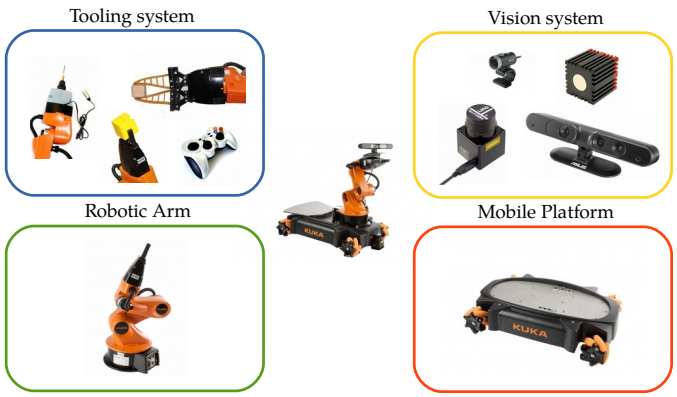


Figure 1.2 **Mobile manipulator subsystems.**

Besides the robotic arm and the mobile platform, an autonomous mobile manipulator is usually equipped with a *vision subsystem*, which is composed by a set of sensor devices necessary to perceive the environment state, and a *tooling subsystem*, whose role is to provide the right actuation instruments to interact with the environment [6] (See Figure 1.2). The most common mobile manipulators, developed in the recent years, are reported in Figure 1.3.

Little Helper (2009)
Department of production
Alborg, Denmark

Justin (2009)
German Aerospace Center
Wessling, Germany

KUKA omniRob (2009)
KUKA laboratories GmbH
Augsburg, Germany

PR2 (2010)
Willow Garage
Menlo Park, California

KUKA youBot (2011)
KUKA laboratories GmbH
Augsburg, Germany

Care-O-Bot (2015)
Fraunhofer-Gesellschaft
Germany

Figure 1.3 **Examples of recent mobile manipulators.**

As described in [3], to achieve a real improvement in the overall productivity, the introduction of mobile manipulators in the industrial field should satisfy logistic, assistive and service requirements. Logistic requirements cover the process of transporting parts between different workstations and storages, that compose the production line, and the process of loading components,

several or one at time, into feeders and machines. Currently, logistics tasks are carried out by humans and represent a critical and expensive process in the production line. Through mobile manipulation, the logistic field can be fully automated, as it has been done, for example, by the well known Amazon company, which has adopted mobile robots to perform autonomous warehouse inventory movements [7]. Assistive tasks cover the processes of loading/unloading materials into machinery for processing, pre-assembling of components, observing and comparing parts to identify and correct defects. Finally, service tasks should assist the production process by maintaining, repairing, overhauling and cleaning the different workstations.
To perform these kind of operations, mobile robots must be perfectly inserted in the ICT system of the company. Through a wireless access point, mobile robots should be connected within the industry network, so that they can communicate with the other information technologies, as the production control and diagnostic system, the Enterprise Resource Planning, the Warehouse Management Systems, as well as with human operators (see Figure 1.4).

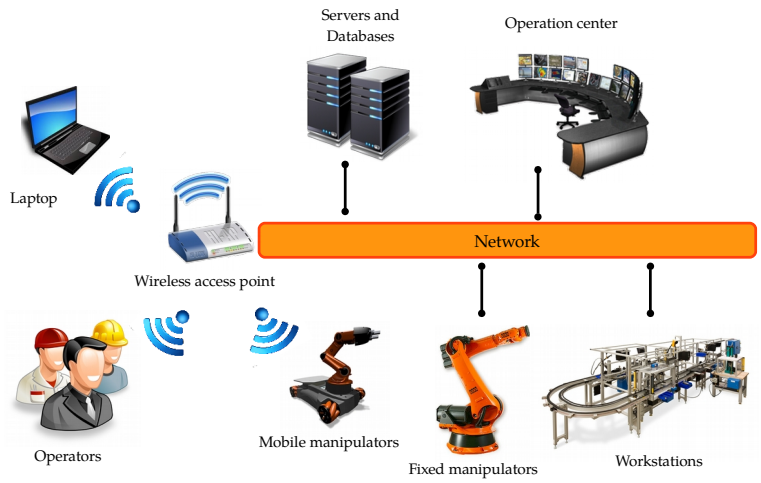


Figure 1.4 **Industrial network architecture.**

In order to fulfil these requirements, navigation and manipulation abilities for mobile manipulators should be developed. Navigation is a fundamental activity for mobile robots, that should be able to safely move from one position of the environment to another one, while avoiding collisions with the

objects possibly disposed in it. In an industrial context, mobile manipulators should navigate from one workstation to another one, while avoiding collision with other robots, machines and human operators. Thus, robots must be provided with a navigation strategy that, based on the visual perceptions provided by cameras and proximity sensors, allows them to move safely inside unknown and dynamic environments. To cope with these issues, often navigation is divided in *local* and *global* navigation [8]. Global navigation, using a map of the environment a-priori known or reconstructed during the motion of the robot, searches for a valid path to reach a desired position. Local navigation, instead, has the purpose of guiding the robot and avoiding collisions with obstacles, based on the perceptions acquired by the vision system in real-time. In this way it is possible to face environment changes, unforeseen by the global navigation.

Mobile manipulators, furthermore, must interact with the environment to perform manipulation tasks, as grasping objects and changing their arrangement in the workspace, assembling parts, opening and closing doors. Nowadays, in the field of industrial robotics, every action composing a manipulation task is first carefully programmed by human operators and then executed by robots, that mechanically repeat it. A behaviour more suitable for autonomous mobile manipulators, would be for the human operators to specify only the final desired configuration of the objects the should be manipulated or assembled and leave to the robot the duty of planning the correct sequence of operations necessary to carry out the manipulation task. Following this perspective, the robot controller should be composed by a *cognitive layer* and a *motion layer.* The cognitive layer, using artificial intelligence techniques [9], should recreate an abstract and symbolic representation of the environment, useful to plan a sequence of actions, that leads to the desired goal configuration of the objects (Figure 1.5). The motion layer, instead, controls the actual motions of the robot by generating the necessary trajectories to accomplish the abstract actions, described by the cognitive layer. To develop this kind of approach, it necessary for the robot to interact autonomously with a great variety of objects, for example it should be able to decide the best way to grasp a certain object starting from its 3D virtual model, and choose how to rearrange it in a new stable position. Moreover, especially for assembly tasks, it is necessary for the robot to interact

with the objects in a compliant way, as in the well known case of the peg in hole insertion [10]. Thus, torque and force control strategies should be considered for a better interaction between the robot and the environment.

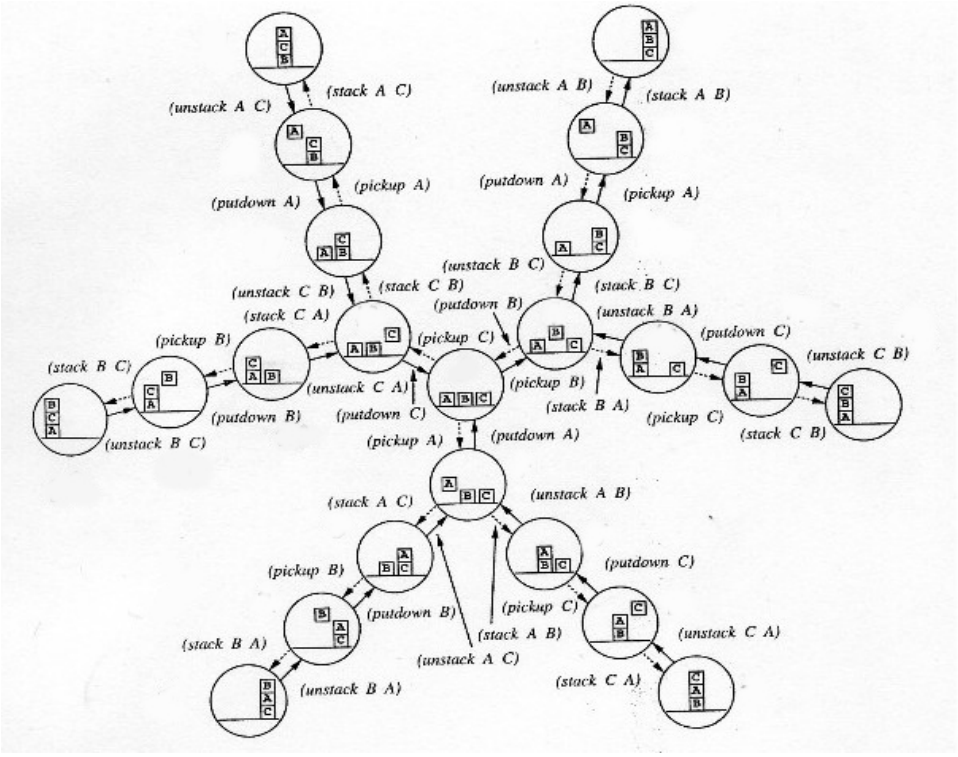
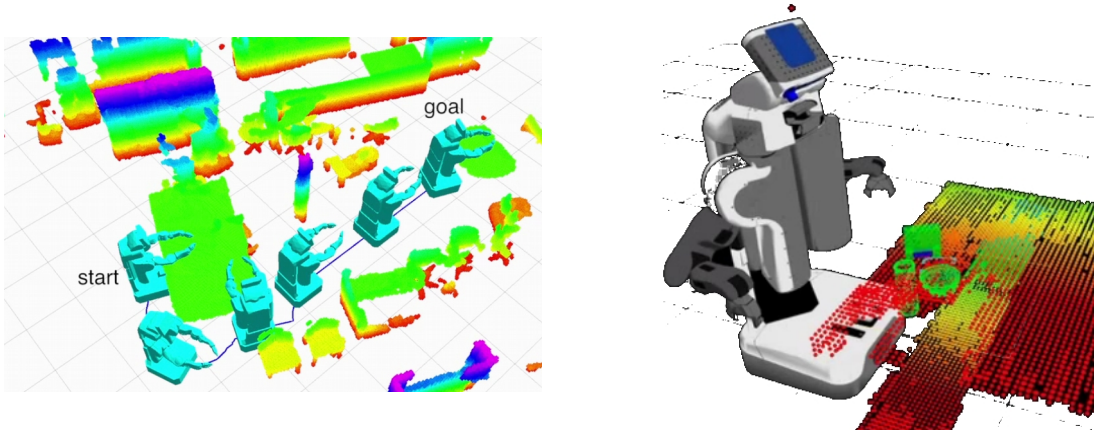


Figure 1.5 Example of symbolic state space for manipulation of three blocks.

Similar navigation and manipulation abilities are needed by service robotics, where mobile manipulators are required to assist human beings, not necessary experts in the robotics or ICT fields, inside everyday environments. Compared to industrial workspace, the autonomous navigation of robots inside houses or offices is even more complex, because they constitute very cluttered and dynamic environments. To guarantee correct and secure motions in these particular environments, a 3D navigation strategy [2] must take into account the whole structure of both the environment and the robot itself to plan collisions free paths (see Figure 1.6). Also manipulation is more critical in the case of service robotics, where robots must execute manipulation tasks on a great variety of completely unknown objects. Indeed, differently from industrial situations, where an accurate and complete model

of the objects to be manipulated is given, in service robotics these models must be reconstructed on the fly by the visual subsystem of the robot. Heuristic strategies, then, should be used to plan and control the manipulation task, based on the reconstructed rough model of the objects (see again Figure 1.6).



3D Navigation Manipulation of unknown objects

Figure 1.6 Example of 3D Navigation and Manipulation of unknown objects with PR2 robot.

Furthermore, a typical issue regarding mobile manipulation is redundancy resolution. A kinematic redundancy occurs when the total degrees of freedom of a robotics system exceed those strictly required to execute a certain task. As mobile manipulators add the degrees of freedom of their mobile base to those of the robotic arm, often they become redundant robot. Since it is widely recognized that a general task consists of following an end-effector Cartesian trajectory using six degrees of freedoms (three for position and three for orientation), a robot with seven or more degree of freedom is considered as the typical example of inherently redundant manipulator. Redundancy allows the robot to execute the same task in infinite different ways. Choosing among all these possibilities an optimal modality to perform a certain operation, is then a critical issue to exploit the wide capabilities of mobile manipulation. The main idea of redundancy resolution is to use the exceeding degrees of freedom to obtain secondary desired behaviours, while the robot executes in a completely consistent way its primary task. Redundancy resolution, for example, can be used to keep the joints values away from their physical limits,

maintain the robot in a configuration where it has a good ability to perform manipulation tasks, avoid obstacles, minimize the energy consumption or support the visual subsystem of the robot.

1.2 Objectives of the thesis

The main objective of this thesis is to study how a mobile robot, equipped with a robotic arm, can execute manipulation tasks in a completely unknown environment. This work, in particular, has the objective of analysing how the redundancy, typical of mobile manipulators, can be used to improve the coordination between mobile base and robotic arm, and how it can be exploited to support and optimize manipulation tasks in general. Furthermore another goal is to develop a navigation technique, which would allow the robot to safely move in the environment while avoiding obstacles.
A final objective is to study how a mobile manipulator can be used, in the context of both industrial and service robotics, to grasp objects, whose shape and position are a-priori unknown.

1.3 Achieved results

In this work, a mobile manipulation framework, composed of control techniques and algorithms, has been developed to perform pick and place operations with the KUKA youBot robotics research platform [11]. KUKA youBot is a robot composed of an omnidirectional mobile platform and a five axis serial manipulator, for a total of eight overall degrees of freedom. The redundancy description of the robot has been introduced at a kinematic level with the definition of a set of redundancy parameters. These redundancy parameters, which describe the physical meaning of the exceeding degrees of freedom of the robot, can be arbitrary set without altering the end-effector configuration. The redundancy of the system has been exploited optimizing the manipulability of the whole mobile manipulator so that configurations of the robot that are more suitable for manipulation tasks are preferred and the coordination between mobile platform and robotic arm results improved. Furthermore redundancy has been utilized to keep the joint values of the manipulator inside their physical limits, to minimize the mobile base movements and to keep some goal object inside the field of view of the vision subsystem.

In order to confer the robot the ability to correctly navigate in the environment, a reactive local control strategy has been proposed, based on the concept of virtual rotational fields, which have the purpose of guiding the robot around the obstacles instead of simply being repulsed by them. In this way the classical local minima problem, characteristic of the potential field navigation technique, has been solved. A set of proximity sensors mounted around the platform or a single depth camera can be equivalently used as a sensory system to acquire the environment information necessary for the navigation of the robot.
Finally an effective and efficient grasp synthesis process has been developed with the purpose of performing manipulation operations in a completely unknown environment, where the information regarding shape, dimension, position and orientation of the objects are not given a-priori to the robot. Thanks to the realized framework, KUKA youBot is able to autonomously explore the environment, reconstruct the model of the objects through a depth camera device, plan and correctly execute pick and place operations. Simulated and experimental validations demonstrate the effectiveness and reliability of our implementation in dealing with manipulation tasks in unknown environments.

1.4 Structure of the thesis

The thesis is organized as follows: the kinematics of the KUKA youBot robot is presented in Chapter 2; possible redundancy resolution strategies are discussed in Chapter 3; in Chapter 4 the developed navigation technique based on rotational potential is illustrated; the realized grasp synthesis process is presented in Chapter 5; implementation details regarding the autonomous pick-and-place framework are illustrated in Chapter 6; experimental results are reported in Chapter 7.

Chapter 2

Kinematics of KUKA youBot mobile manipulator

In this work, a mobile manipulator is considered as a system composed of a robotic arm mounted on a mobile base. Often these two components are considered as separate elements and not as a whole system. In this scenario first the mobile base is used only with navigation purposes, e.g. to approach a certain object, and then the robotic arm performs the manipulation task as if it was fixed to the ground. This approach does not take into account the whole capabilities of the system, as not all its degrees of freedom are used simultaneously. A more efficient way to accomplish a manipulation task is for the base and the arm to simultaneously cooperate. Considering the synergies between the mobile base and the manipulator is a better way to tackle the problem of mobile manipulation to exploit the high level of dexterity it offers. For this reasons the kinematic model of a mobile manipulator must describe the characteristic of both the base and the arm in a comprehensive way. Moreover mobile manipulators, due to their high number of DOFs, are often redundant system. Thus, introducing a description of their redundancy at kinematics level can be an efficient measure to exploit the specific geometric characteristics of a certain robot.

In this chapter a kinematic model of the KUKA youBot mobile manipulator will be presented. First the forward kinematics will be introduced, then the specific redundancy parameters of the robot and their geometric meaning will be explained, making it possible to define a closed form inverse kinematics algorithm. Finally a condition of existence for the inverse kinematics solution and the ranges of existences of the redundancy parameters will be discussed.

2.1 Forward kinematics

The state of a mobile manipulator can be expressed in a general form as:

$$\boldsymbol{q}=\begin{bmatrix}\boldsymbol{q_b}\\ \boldsymbol{q_a}\end{bmatrix} \tag{2.1}$$

where subscript b is assigned to the base variables $\boldsymbol{q_b}=[X_b\, Y_b\, Z_b\, \alpha_b \beta_b \gamma_b]^T$, that represent position and orientation of the mobile base, while subscript a denotes the arm joint position variables $\boldsymbol{q_a}=[q_1\, q_2 \ldots\, q_n]^T$ of a manipulator with n degrees of freedom. In particular, the variables X_b, Y_b and Z_b define the absolute position of the mobile base centre of mass, while α_b, β_b and γ_b are respectively the raw, pitch and yaw angles that determine the base orientation.

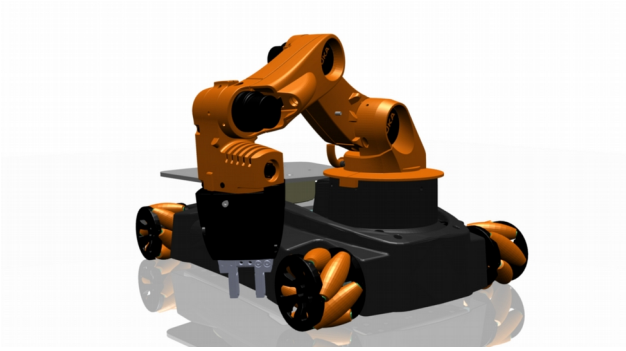


Figure 2.1 **KUKA youBot mobile manipulator.**

The KUKA youBot (Figure 2.1) is a robot composed of an omnidirectional mobile platform and a five axis serial manipulator. For a complete hardware specification of the robot refer to Appendix A.

Since the mobile platform is capable only of planar movement and only the case of a single robotic arm mounted on the base is considered, the state of KUKA youBot will be defined as

$$\boldsymbol{q_b}=[X_b\; Y_b\; \theta_b]^T \tag{2.2}$$

$$\boldsymbol{q_a}=[q_1\; q_2\; q_3\; q_4\; q_5]^T \tag{2.3}$$

where θ_b is the planar absolute orientation of the mobile platform (see Figure 2.2).

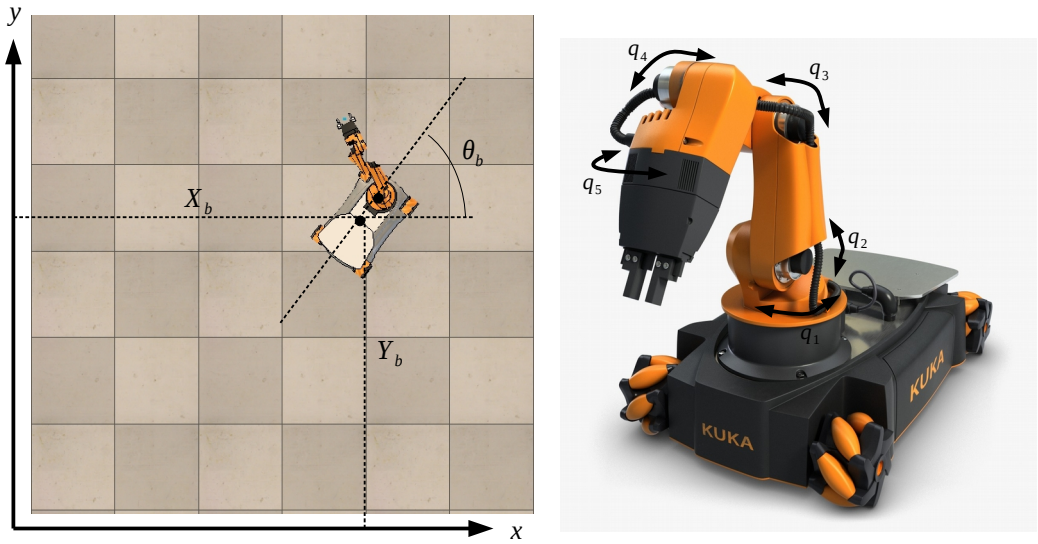


Figure 2.2 **KUKA youBot state variables**

Forward kinematics is the basic mathematical tool for finding the Cartesian position and orientation of the robot end-effector knowing its state variables. Direct kinematics for a mobile manipulator can be expressed in a general form through the following homogeneous transformation matrix:

$$K_F(\boldsymbol{q}) = T_b T_d T_a \tag{2.4}$$

where T_b is the homogeneous transformation matrix from the absolute frame to the mobile base frame, T_d is a constant transformation matrix that denotes the displacement between the platform frame and the arm base frame, T_a is the homogeneous transformation matrix from the arm base frame to the end-effector frame, depending on the manipulator kinematics chain configuration. For the particular case of the KUKA youBot robot the previous kinematics terms assume the form:

$$T_b = \begin{bmatrix} \cos(\theta_b) & -\sin(\theta_b) & 0 & X_b \\ \sin(\theta_b) & \cos(\theta_b) & 0 & Y_b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

$$T_d = \begin{bmatrix} 1 & 0 & 0 & x_d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$T_a = A_1^0(q_1) A_2^1(q_2) A_3^2(q_3) A_4^3(q_4) A_5^4(q_5) \tag{2.7}$$

where

$$A_i^{i-1}(q_i) = \begin{bmatrix} c_{q_i} & -s_{q_i} c_{\alpha_i} & s_{q_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{q_i} & c_{q_i} c_{\alpha_i} & -c_{q_i} s_{\alpha_i} & a_i s_{q_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

follows the standard Denavit-Hartenberg convention [12].

2.2 Redundancy parameters

A kinematic redundancy [13] occurs when the total DOFs of a robotics systems exceed those strictly required to execute a certain task. Therefore robots are not inherently redundant, rather there are tasks with respect to which they may become redundant. Since it is widely recognized that a general task consists of following an end-effector Cartesian trajectory using six degrees of freedoms (three for position and three for orientation), a robot with seven or more degree of freedom is considered as the typical example of *inherently redundant* manipulator. However even robot with fewer degrees of freedom may become kinematically redundant for specific tasks, presenting

task redundant behaviour. Redundancy can even be noticed in particular cases when there are more then one suitable joints configurations for reach a given end-effector pose, as for example in the well know case of high or low elbow in an anthropomorphic arm manipulator (figure 2.3).

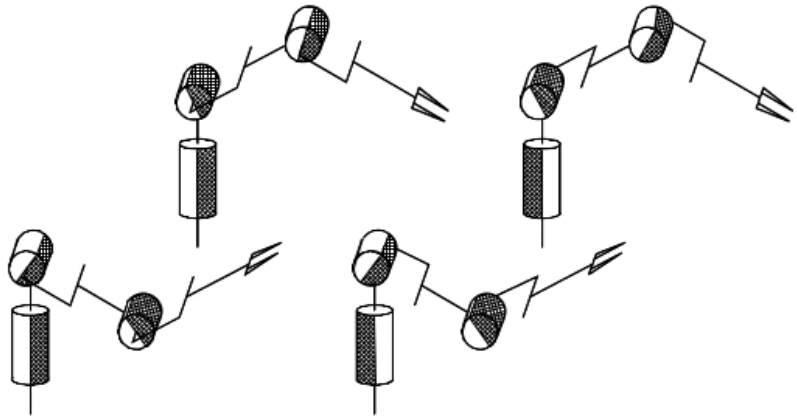


Figure 2.3 The four configurations of an anthropomorphic arm compatible with a given wrist position.

Image taken from B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo: "Robotics - Modelling, planning and control", Springer, 2009, p. 99.

Redundancy is a key feature for manipulation actions. In fact, additional degrees of freedom besides those strictly required to execute a certain end-effector task can be utilized to avoid singularities, joint limits, workspace obstacles, but also to minimize joint torque, energy or, in general, optimize suitable performance indexes.
Redundancy has a strong connection with the inverse kinematics problem that can be summarized as finding the joints values that place the end-effector in a given Cartesian position and orientation:

$$\boldsymbol{q}: \ \boldsymbol{k}_F(\boldsymbol{q}) = \boldsymbol{x} \tag{2.9}$$

where $\boldsymbol{k}_F: \ \Re^n \rightarrow \Re^m$ represents the forward kinematics function that maps the n joint variables into the desired end-effector pose $\boldsymbol{x} = [x_e \ y_e \ z_e \ \alpha_e \ \beta_e \ \gamma_e]$, composed by the position variables x_e , y_e , z_e

and the Tait-Bryan angles α_e , β_e , γ_e that define the end-effector orientations. The dimensional number m corresponds to the number of degrees of freedom necessary to accomplish a certain Cartesian task, and generally is equal to six.

In case of redundant robot, since the condition $n > m$ is always verified, the inverse kinematics problem is underconstrained, therefore a *Redundancy Resolution* strategy to fully constrain the problem must be defined.

Redundancy resolution is a well known issue in control engineering and many different strategies have been proposed over the years, for example *Extended Jacobian*, *Projected Gradient*, *Reduced Gradient* [14]. These techniques are based on the assumption that a closed form inverse kinematics is not available due to the difficulty of solving such analytical problem for redundant manipulator, so first-order differential kinematics [15] is taken in consideration as the main mathematical tool for control strategies. These methods provide a good level of abstraction as they possess the property of being general and robot independent.

Another way to approach redundancy resolution is to study the specific geometry and kinematics of a robot and extract the redundancy parameters that provide a physical meaning to the exceeding degrees of freedom. Although this method is not robot independent as the others, it offers the possibility to exploit the redundancy of the system in a much more efficient way, because each redundancy parameter can be used to optimize a specific behavior. Moreover the expensive computations needed for differential kinematics are not necessary with this method. The Redundancy parameters are used to describe the robot internal motion, namely a particular variation of the joint values that does not entail a variation of the end-effector position and orientation.

Let $\boldsymbol{\rho}=[\rho_1\ \rho_2\ ...\ \rho_r]^T$ be a vector of redundancy parameters with dimension $r = n-m$. The fully constrained inverse kinematics problem can now be formulated as:

$$\boldsymbol{k_I}(\boldsymbol{x},\boldsymbol{\rho})=\boldsymbol{q} \tag{2.10}$$

where $\boldsymbol{k_I}$: $\Re^{m+r} \rightarrow \Re^n$ is an inverse kinematics function that computes the joints values needed to assume a goal pose $\boldsymbol{x}$ with redundancy parameters $\boldsymbol{\rho}$.

The KUKA youBot robot possesses eight overall degrees of freedom, three provided by the mobile base and five by the arm, hence it can be classified as an inherently redundant mobile manipulator. A set of redundancy parameters and the relative closed form inverse kinematics has been proposed for KUKA youBot in [16], where two redundancy parameters, ρ_1 and ρ_2 , are introduced for fully constrain the two exceeding degree of freedom, while parameter ρ_3 discriminates the high or low elbow configurations of the arm. In the following section the redundancy parameters of KUKA youBot will be presented in detail, as well as their physical meaning and their relation with the state variables.

The redundancy parameter ρ_1 denotes the angular displacement between the mobile platform and the manipulator. The redundancy can be described as the ability of the base to perform a rotation around the first axis of the manipulator leaving the arm fixed in its position (Figure 2.4). Furthermore parameter ρ_1 can be expressed as a function of the state variables through the simple relation:

$$\rho_1 = q_1 \quad . \tag{2.11}$$

From the previous equation it can be easily seen that parameter ρ_1 is subjected to the joint limit of the first axis of the manipulator: $q_1^{min} < \rho_1 < q_1^{max}$.

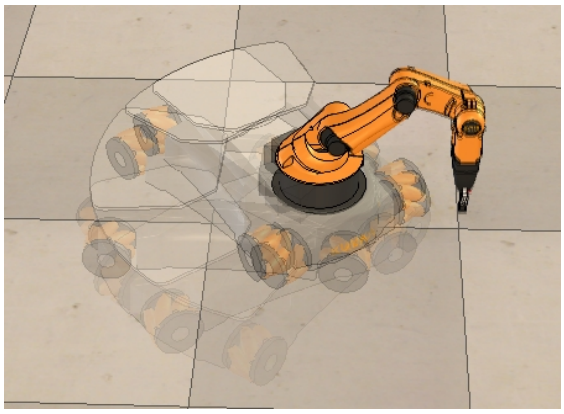


Figure 2.4 Internal motion induced by ρ_1 redundancy parameter.

Parameter ρ_2 models the extension of the youBot arm. Given an end-effector pose, in fact, the whole mobile manipulator can be more or less outstretched, placing the platform at different distances from the end-effector goal pose (Figure 2.5). Parameter ρ_2 relation with the state variables is given by:

$$\rho_2 = a_2 \sin(q_2) + a_2 \sin(q_2 + q_3) + d_5 \sin(q_2 + q_3 + q_4) \tag{2.12}$$

where a_2 , a_3 and d_5 are the Denavit-Hartenberg parameters characteristic of the manipulator geometry (see Appendix A). The range of existence of parameter ρ_2 depends on the desired pose of the end effector and thus it will be discussed in detail in section **2.4** after the inverse kinematics algorithm.

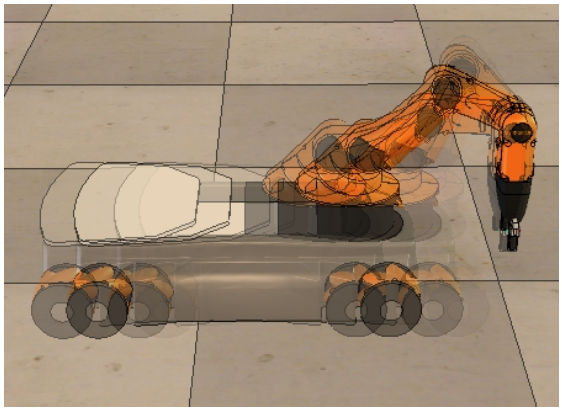


Figure 2.5 Internal motion induced by ρ_2 redundancy parameter.

Parameter ρ_3 denotes the two possible configurations, high or low elbow, that emerge from the arm inverse kinematics resolution (Figure 2.6).

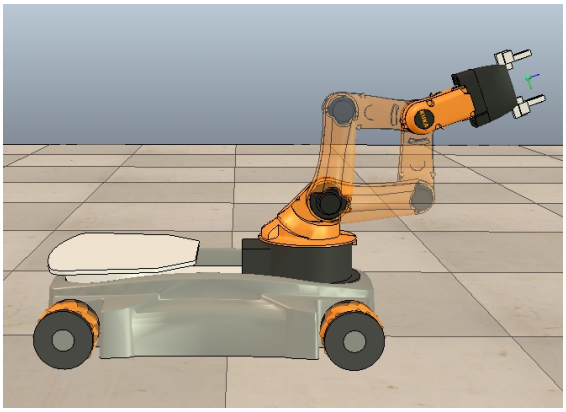


Figure 2.6 Elbow up or down configurations induced by parameter ρ_3 .

Besides the three redundancy parameters presented in [16], an additional parameter ρ_4 is introduced in this work. This parameters is exploited to solve the redundancy situation depicted in Figure 2.7, where the end-effector rotational axis is parallel to the vertical axis z , and that will be referred to as "vertical configuration redundancy".

In particular, redundancy parameter ρ_4 defines the absolute orientation of the robotics arm, which can be arbitrarily set for any vertical configuration of the end-effector, because the last joint of the manipulator can be used to compensate the rotational displacement of the whole robot (Figure 2.7).

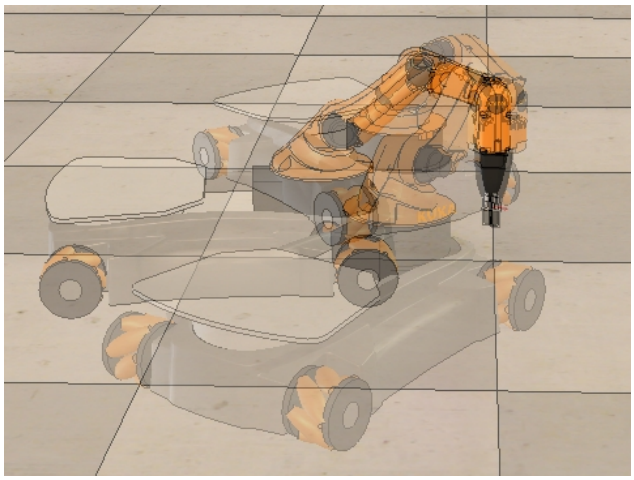


Figure 2.7 Task redundancy induced by parameter ρ_4 .

The introduction of parameter ρ_4 is particularly important, because vertical configurations are the most intuitive ones to be used with this kind of robot for pick and place operations.
Parameter ρ_4 can be expressed as function of the robot state variables through the following relation:

$$\rho_4 = q_1 + \theta_b \quad . \tag{2.13}$$

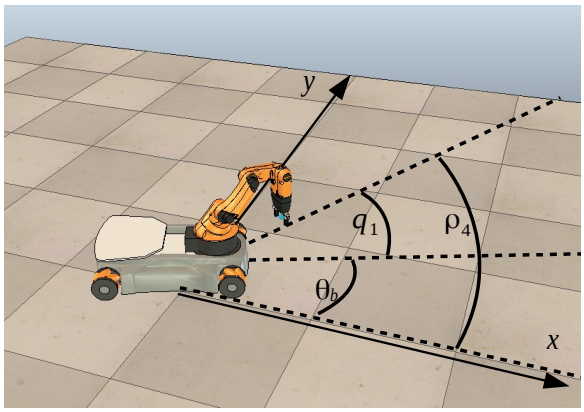


Figure 2.8 Geometric meaning of redundancy parameter ρ_4 .

2.3 Inverse Kinematics

As stated before, inverse kinematics consists in the fundamental process of finding the joints values that grant a desired end-effector pose.
A general analytical algorithm to accomplish such a task does not exist, therefore inverse kinematics must be derived for each robot through a geometric and algebraic specific inspection. In the following section a geometric procedure to retrieve the inverse kinematics for KUKA youBot will be presented.

In [16] the authors first resolve the inverse kinematics for the arm according to redundancy parameter ρ_2 and ρ_3 , then compute the correspondent configuration of the mobile platform taking in consideration ρ_1 . The algorithm proposed in this work follows this methodology, although the task redundancy described through ρ_4 is introduced at kinematics level and

greater attention is given to the conditions of solution existence and the redundancy parameters admissible ranges.

The objective of the youbot inverse kinematics algorithm is to find that particular joint configuration $\boldsymbol{q}=[X_b\ Y_b\ \theta_b\ q_1\ q_2\ q_3\ q_4\ q_5]^T$ that realises the six dimensional goal pose of the end-effector $\boldsymbol{X_g}=[x_g\ y_g\ z_g\ \alpha_g\ \beta_g\ \gamma_g]^T$ when redundancy parameters $\boldsymbol{\rho}=[\rho_1\ \rho_2\ \rho_3\ \rho_4]^T$ are specified.
First the end-effector reference frame is extracted from the goal Tait-Bryan angles (Figure 2.9).

$$\boldsymbol{Z_e}=\begin{bmatrix} z_{e_1} \\ z_{e_2} \\ z_{e_3} \end{bmatrix}=\begin{bmatrix} \sin(\beta_g) \\ -\cos(\beta_g)\sin(\alpha_g) \\ \cos(\alpha_g)*\cos(\beta_g) \end{bmatrix} \tag{2.14}$$

$$\boldsymbol{X_e}=\begin{bmatrix} x_{e_1} \\ x_{e_2} \\ x_{e_3} \end{bmatrix}=\begin{bmatrix} \cos(\beta_g)\cos(\gamma_b) \\ \cos(\alpha_g)*\sin(\gamma_g)+\cos(\gamma_g)*\sin(\alpha_g)*\sin(\beta_g) \\ \sin(\alpha_g)*\sin(\gamma_g)-\cos(\alpha_g)*\cos(\gamma_g)*\sin(\beta_g) \end{bmatrix} \tag{2.15}$$

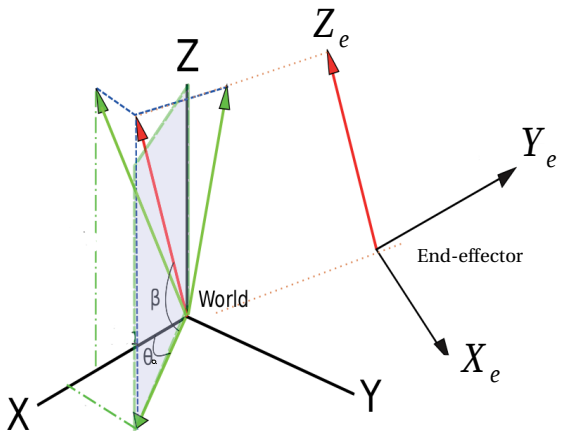


Figure 2.9 **End-effector reference frame with respect to the world absolute frame.** Image taken from [16].

The vertical inclination β of the end-effector can be expressed as:

$$\beta = atan2\left(z_{e_3}, \sqrt{z_{e_1}^2 + z_{e_2}^2}\right) \tag{2.16}$$

The absolute planar orientation of the end-effector θ_0 results as:

$$\theta_0 = atan2\left(z_{e_2}, z_{e_1}\right) \tag{2.17}$$

The absolute orientation of the whole youBot arm corresponds to θ_0 when the end-effector is not in a vertical configuration. When the vertical configuration occurs, ρ_4 is used to define θ_0, and so, in this work, Equation (2.17) is modified as follows:

$$\theta_0 = \begin{cases} \rho_4 & if \quad \mathbf{Z_e} // [001]^T \\ atan2\left(z_{e_2}, z_{e_1}\right) & otherwise \end{cases} \tag{2.18}$$

At this point it is possible to solve the inverse kinematics for joints q_2, q_3 and q_4, as they represent a standard three link planar manipulator, which is constrained to lay in the vertical plane identified by θ_0 (Figure 2.10).

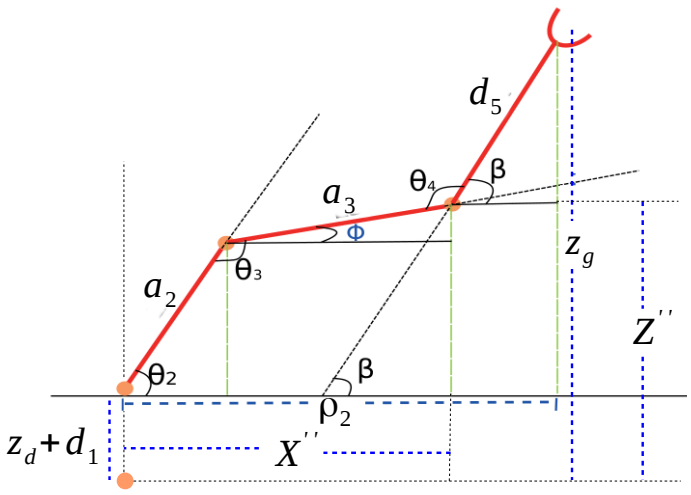


Figure 2.10 **Link 2, 3 and 4 of the KUKA youBot manipulator.** Image taken from [16].

The following relations hold:

$$Z'' = z_g - z_d - d_1 - d_5 \sin(\beta) \tag{2.19}$$

$$X'' = \rho_2 - d_5 \cos(\beta) \tag{2.20}$$

Values for the joints position, accounting for redundancy parameters ρ_2 and ρ_3, are obtained through the procedure:

a) $$\cos(\theta_3) = \frac{(-Z''^2 - X''^2 + a_2^2 + a_3^2)}{(2 a_2 a_3)} \tag{2.21}$$

b) $$\sin(\theta_3) = sign(\rho_3)\sqrt{1 - \cos(\theta_3)^2} \tag{2.22}$$

c) $$q_3 = atan2(\sin(\theta_3), \cos(\theta_3)) - \pi \tag{2.23}$$

d) $$k_1 = a_2 - a_3 \cos(\theta_3) \tag{2.24}$$

e) $$k_2 = a_3 \sin(\theta_3) \tag{2.25}$$

f) $$q_2 = atan2(Z'', X'') + atan2(k_2, k_1) - \frac{\pi}{2} \tag{2.26}$$

g) $$q_4 = \beta - q_2 - q_3 - \frac{\pi}{2} \tag{2.27}$$

The rotational displacement θ_5 of the last link, necessary to align the end-effector with the desired pose (Figure 2.11), can be computed as:

h) $$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{21} & e_{22} & e_{23} & e_{24} \\ e_{31} & e_{32} & e_{33} & e_{34} \\ e_{41} & e_{42} & e_{43} & e_{44} \end{bmatrix} = A_1^0(\theta_0) A_2^1(q_2) A_3^2(q_3) A_4^3(q_4) A_5^4(0) \tag{2.28}$$

i) $$\cos(\theta_5) = e_{11} x_{e_1} + e_{21} x_{e_2} + e_{31} x_{e_3} \tag{2.29}$$

j) $$\sin(\theta_5) = e_{12} x_{e_1} + e_{22} x_{e_2} + e_{32} x_{e_3} \tag{2.30}$$

k) $$q_5 = atan2(\sin(\theta_5), \cos(\theta_5)) \tag{2.31}$$

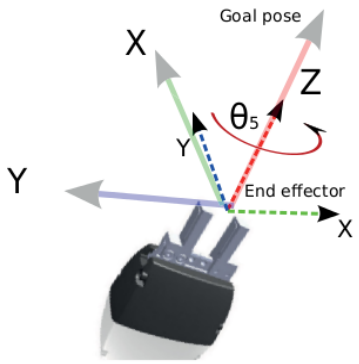


Figure 2.11 **Rotational displacement of the last link.** Image taken from [16].

As stated before the value of q_1 corresponds to the rotational displacement between base and manipulator, hence:

l) $$q_1 = \rho_1 \quad . \tag{2.32}$$

As the configuration of the manipulator has been completely determined, the position and orientation of the mobile platform, considering the redundancy of parameter ρ_1 , can be easily derived:

$$\theta_b = \theta_0 - \rho_1 \tag{2.33}$$

$$X_b = x_g - e_{14} - x_d \cos(\theta_b) \tag{2.34}$$

$$Y_b = y_g - e_{24} - x_d \sin(\theta_b) \tag{2.35}$$

2.4 Conditions for the existence of a solution

The existence of a solution for the inverse kinematics problem, as for every nonlinear systems, is not always guaranteed. In the general case, a solution for inverse kinematics exists if the desired end-effector pose lies in the *dexterous workspace*, the volume of space which the robot end-effector can reach with every orientation. Furthermore, in the particular case of this work, as the kinematics has been constrained with redundancy parameters, the solution

existence depends even on the values of these parameters, which must lie in a certain range defined by their physical meaning.
In this section a set of conditions to grant the existence of the inverse kinematics solution for the KUKA youBot will be proposed.

In the case of a mobile manipulator, since the base is free to move in the environment, the dexterous workspace has limitations only in the height of the goal pose, that can not be too high.
A condition of existence for the KUKA youBot inverse kinematics can be derived considering the relation:

$$-1<\cos(\theta_3)<+1 \tag{2.36}$$

that can be expanded through Equation (2.21) as:

$$-1<\frac{(-Z''^2-X''^2+a_2^2+a_3^2)}{(2\,a_2\,a_3)}<+1 \quad . \tag{2.37}$$

Solving for ρ_2 , after some algebraic passages, the following relation can be found:

$$t_1-t_2<(\rho_2-t_3)^2<t_1+t_2 \tag{2.38}$$

where $t_1=2\,a_2\,a_3$, $t_2=a_2^2+a_3^2-Z''^2$ and $t_3=d_5\cos(\beta)$.

From the previous inequality, it is possible to infer two important results: a condition of membership to the dexterous workspace, and a range of admissibility for parameter ρ_2 .

Dexterous workspace condition:

$$t_1+t_2>0 \tag{2.39}$$

Arm extension range:

$$t_3\pm\sqrt{t_1-t_2}<\rho_2<t_3\pm\sqrt{t_1+t_2} \tag{2.40}$$

If both conditions are satisfied, a solution to the inverse kinematics problem for the KUKA youBot is guaranteed to exist. As expected, the dexterous workspace condition depends only on the height of the goal pose, and so defines the maximum reachable height, given a certain end-effector vertical orientation. The arm extension range defines the minimum and maximum extension values that the arm can assume in a given end-effector configuration. These conditions are very useful and must be taken into consideration for the implementation of the inverse kinematics algorithm to make it more robust. For example, if the height of the desired goal position is outside the workspace, the dexterous workspace condition can be used to set the robot end-effector near the workspace limits, so that as soon as the goal becomes reachable the robot will be as close as possible to it.
These conditions will be resumed in this work in the chapter devoted to grasp synthesis, as they will be used to identify unfeasible grasp configurations.

Chapter 3

Redundancy Resolution

Redundant robots present more degrees of freedom than those strictly required to accomplish a certain task. The main idea of redundancy resolution is to use the exceeding degrees of freedom to obtain secondary desired behaviours [13] [17], while the robot executes in a completely consistent way its primary task. Hence the redundancy resolution problem consists in designing optimization criteria, that allow to achieve some subsidiary behaviours, as for example keeping the joints values away from their physical limits, maintaining the robot in a configuration where it has a good ability to perform manipulation tasks, avoiding obstacles or minimizing the energy consumption.
Having defined the robot redundancy through a set of redundancy parameters, which can be arbitrarily set without altering the primary task execution, it is possible to account for these optimization criteria in a specific way, namely each parameter can be utilized to achieve a particular behaviour based on its physical meaning. In this work, an objective function has been designed for every redundancy parameter considering a desired optimization criterion. Each redundancy parameter value, necessary for the solution of the inverse kinematics problem, is then chosen in order to maximize its respective objective function. In this chapter, possible redundancy resolution strategies for each redundancy parameter of KUKA youBot will be discussed.

3.1 Arm extension redundancy

The first objective of any redundancy resolution strategy must be to preserve the robot primary task. For example, the ideal kinematics model introduced in the previous chapter does not take into account the feasible range of the joints variables, resulting in possibly not achievable solutions of the inverse kinematics problem. The arm extension redundancy, described by parameter ρ_2 for KUKA youBot, can then be used to keep the manipulator joints away

from their physical limits. For each joint of the KUKA youBot, the physical limits can be expressed as:

$$\forall i:\ q_i^{min} < q_i < q_i^{max},\ i = 1...5 \tag{3.1}$$

where q_i^{min} and q_i^{max} represent the lower and upper joint limits.
Given an arm joints variables vector $\boldsymbol{q_a}$, the distance from joints physical limits can be defined by the objective function:

$$U_l(\boldsymbol{q_a}) = 1 - \max_{1 \le i \le 5} \frac{2\left| \frac{q_i^{max} + q_i^{min}}{2} - q_i \right|}{|q_i^{max} - q_i^{min}|} \tag{3.2}$$

Function U_l increases with respect to the distance from the joints limits and is positive only if all the joint variables are inside their range.
Exploiting the arm extension redundancy only to avoid the joint limits would be too simplistic, as the constraint on the joint ranges is an objective quite easy to achieve. For this reason, as suggested in [18], the arm extension redundancy can be also used to maximize the robot manipulability. Given the robot Jacobian

$$J(\boldsymbol{q}) = \frac{\partial \boldsymbol{x_p}}{\partial \boldsymbol{q}} \tag{3.3}$$

where $\boldsymbol{x_p}$ is the end-effector Cartesian state vector, composed by the positional and orientation variables, the manipulability measure [19] can be defined as:

$$U_m(\boldsymbol{q}) = \sqrt{|J(\boldsymbol{q}) J(\boldsymbol{q})^T|}\ . \tag{3.4}$$

The manipulability is a measure of the ability of the robot to modify in any direction the end-effector pose with respect to its current joint configuration. Choosing the joint configuration that maximizes the manipulability for a

desired end-effector pose would then allow easier possible modification of the end-effector pose in the workspace.
In [18] the authors utilize a manipulability measure that takes into account only the degrees of freedom of the manipulator, forcing the robot to take configurations with a short extension of the arm, that often collides with the mobile platform. To exploit the potentiality of mobile manipulation, a viable solution is to consider the manipulability of the whole system. For this reason the end-effector position of a mobile manipulator can be expressed in a general way as:

$$\boldsymbol{p_e} = \boldsymbol{p_b} + R_b[\,\boldsymbol{p_d} + \boldsymbol{k}_{F_a}(\boldsymbol{q_a})] \tag{3.5}$$

where $\boldsymbol{p_b}$ is the position of the mobile base, R_b is a rotational matrix that describes the orientation of the base with respect to the absolute reference frame, $\boldsymbol{p_d}$ is the displacement between the mobile base and the base of the arm, $\boldsymbol{k}_{F_a}(\boldsymbol{q_a})$ is the arm forward kinematics. If the 3D base position and orientation variables are decomposed in translational ones as $\boldsymbol{q}_{b,T} = [\,X_b\ Y_b\ Z_b]^T$ and rotational ones as $\boldsymbol{q}_{b,R} = [\,r_b\ p_b\ y_b]^T$, then $\boldsymbol{p_b}$ depends only on the translational component of $\boldsymbol{q_b}$ and R_b only on the rotational ones. At this point the Jacobian matrix of the whole mobile manipulator, necessary for the manipulability measure computation, can be defined as:

$$J(\boldsymbol{q}) = \frac{\partial \boldsymbol{p_e}}{\partial \boldsymbol{q}} = \left[\frac{\partial \boldsymbol{p_e}}{\partial \boldsymbol{q}_{b,T}} \quad \frac{\partial \boldsymbol{p_e}}{\partial \boldsymbol{q}_{b,R}} \quad \frac{\partial \boldsymbol{p_e}}{\partial \boldsymbol{q_a}}\right] = \left[I \quad \frac{\partial R_b}{\partial \boldsymbol{q}_{b,R}}[\boldsymbol{p_d} + \boldsymbol{k}_{F_a}(\boldsymbol{q_a})] \quad R_b J_a(\boldsymbol{q_a})\right] \tag{3.6}$$

where I is the identity matrix, and $J_a(\boldsymbol{q_a})$ is the Jacobian matrix of the arm. The previous equation shows a simple and efficient way to compute the whole mobile manipulator Jacobian starting from the variables of the base and the arm as they were two decoupled systems. This structure could be very useful because often the terms $\boldsymbol{k}_{F_a}(\boldsymbol{q_a})$ and $J_a(\boldsymbol{q_a})$ are already computed in many manipulator control schemes.

The manipulability measure for the whole KUKA youBot mobile manipulator, for example, can be derived in accordance with the previous relations partitioning the state variables as $\boldsymbol{q}_{b,T}=[\,X_b\;Y_b]^T$, $\boldsymbol{q}_{b,R}=[\,\theta_b]^T$, $\boldsymbol{q}_a=[q_1\;q_2\;q_3\;q_4\;q_5]^T$. The end-effector position is then defined as

$$\boldsymbol{p_e}=\begin{bmatrix}X_b\\Y_b\\0\end{bmatrix}+\begin{bmatrix}\cos(\theta_b)&-\sin(\theta_b)&0\\\sin(\theta_b)&\cos(\theta_b)&0\\0&0&1\end{bmatrix}\left(\begin{bmatrix}x_d\\0\\z_d\end{bmatrix}+\boldsymbol{k}_{F_a}(\boldsymbol{q_a})\right) \tag{3.7}$$

and the terms composing the Jacobian matrix can be computed as

$$\frac{\partial \boldsymbol{p_e}}{\partial \boldsymbol{q}_{b,T}}=\begin{bmatrix}1&0\\0&1\\0&0\end{bmatrix} \tag{3.8}$$

$$\frac{\partial \boldsymbol{p_e}}{\partial \boldsymbol{q}_{b,R}}=\begin{bmatrix}-\sin(\theta_b)&-\cos(\theta_b)&0\\\cos(\theta_b)&-\sin(\theta_b)&0\\0&0&1\end{bmatrix}\left(\begin{bmatrix}x_d\\0\\z_d\end{bmatrix}+\boldsymbol{k}_{F_a}(\boldsymbol{q_a})\right) \tag{3.9}$$

$$\frac{\partial \boldsymbol{p_e}}{\partial \boldsymbol{q_a}}=\begin{bmatrix}\cos(\theta_b)&-\sin(\theta_b)&0\\\sin(\theta_b)&\cos(\theta_b)&0\\0&0&1\end{bmatrix}J_a(\boldsymbol{q_a}) \quad . \tag{3.10}$$

The global objective function, accounting for the constraints on the joints limits and the optimization of the manipulability measure, is composed as:

$$U(\boldsymbol{q})=\begin{cases}U_l(\boldsymbol{q_a}) & \textit{if } U_l(\boldsymbol{q_a})<0\\U_m(\boldsymbol{q})P(\boldsymbol{q_a}) & \textit{otherwise}\end{cases} \tag{3.11}$$

where $P(\boldsymbol{q_a})$ is a penalization factor [20] that considers the joint limits and ensures the continuity of the objective function between its two cases. The adopted penalization factor is equal to zero if the joints are near their limits and to one otherwise, ant it is defined as

$$P(\boldsymbol{q_a}) = 1 - \exp\left(-k \prod_{i=1}^{5} \frac{(q_i - q_i^{min})(q_i^{max} - q_i)}{(q_i^{max} - q_i^{min})^2}\right) \quad , \tag{3.12}$$

where the positive parameter k is a scaling factor.

The optimization over the redundancy parameter ρ_2 to maximize the objective function $U(\boldsymbol{q})$ can be executed through the well known gradient ascent method [21]. This on-line technique allows to perform the optimization, based on the robot current state: for each time instant the values for ρ_2 is determined as

$$\rho_2^{t+\Delta t} = \rho_2^t + \gamma \frac{\partial U(\boldsymbol{q})}{\partial \rho_2} \tag{3.13}$$

where γ is a positive parameter used for tuning the speed and accuracy of convergence of the gradient method. The current value of ρ_2 can be retrieved from the manipulator state $\boldsymbol{q_a}$ through Equation (2.12).

The behavior of the objective $U(\boldsymbol{q})$ as function of parameter ρ_2 is shown in Figure 3.1 and 3.2 for two end-effector configurations. The red lines determine the intervals of ρ_2 where $U(\boldsymbol{q})$ is positive and so all the joint values are inside their limits. Note that these feasible ranges are quite tight, especially in lateral end-effector configurations, a situation that occurs when the end-effector is parallel to the floor, as shown in Figure 3.2.

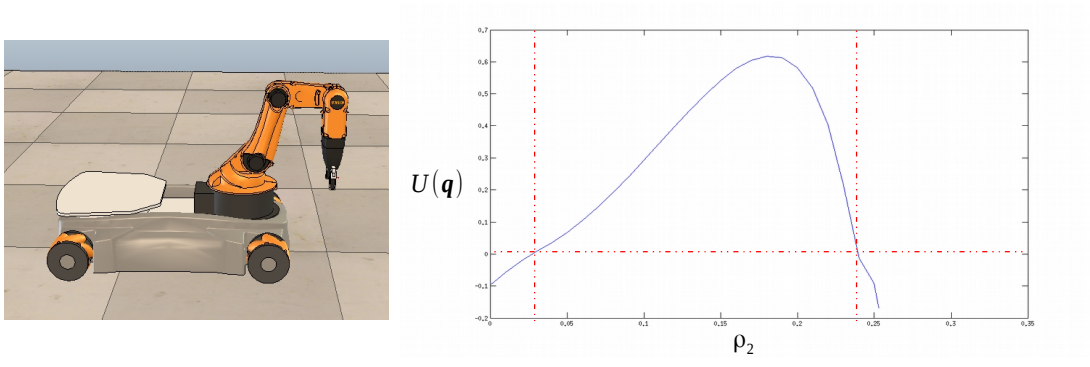


Figure 3.1 Behaviour of $U(\boldsymbol{q})$ as function of ρ_2 for a vertical configuration.

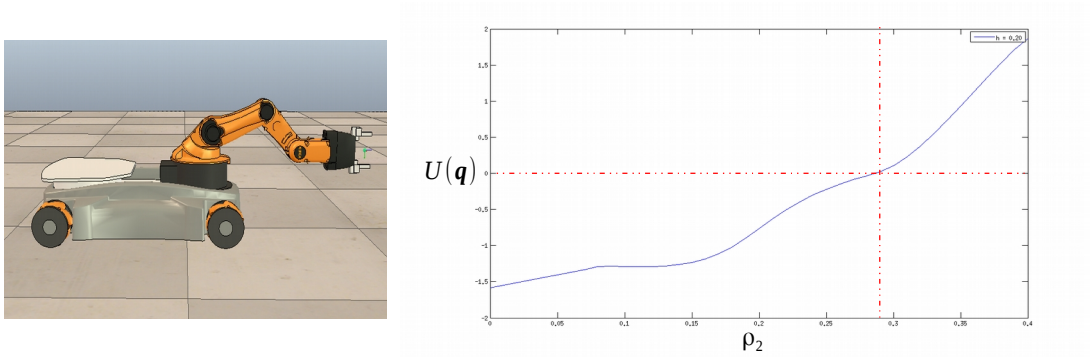


Figure 3.2 Behaviour of $U(\boldsymbol{q})$ as function of ρ_2 for a lateral configuration.

3.2 Elbow redundancy

The redundancy parameter ρ_3 determines the configuration of the elbow: positive values of ρ_3 imply the high elbow, while negative ones the low elbow posture. To be precise this kind of situations, where the inverse kinematics presents multiple but finite solutions, can not be classified as a pure redundancy as the other ones, however it has been handled with the parameter ρ_3 because it is necessary for a complete definition of the KUKA youBot inverse kinematics problem.

As the elbow down configuration causes in many common circumstances a collision between the arm and the mobile platform or results in unfeasible poses due to the physical limits of the joints, the elbow up configuration has been preferred in this work.

3.3 Vertical posture redundancy

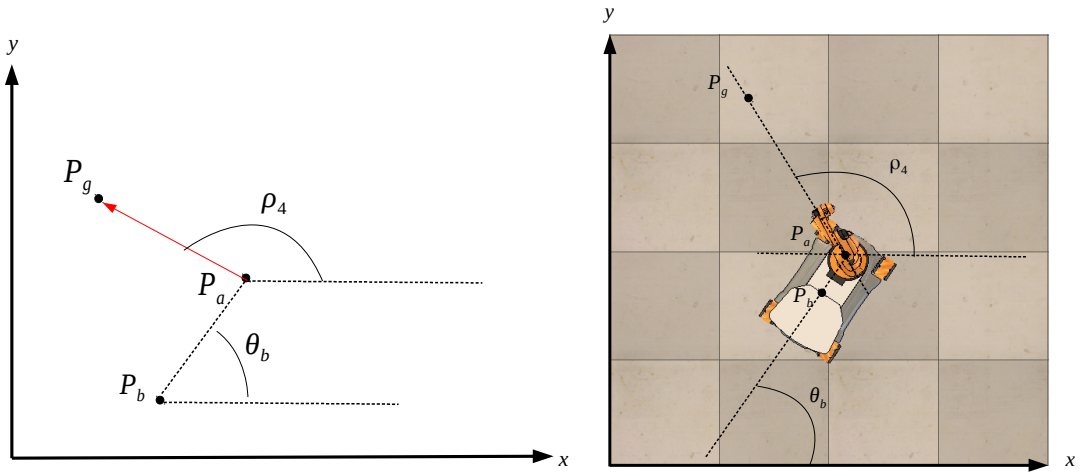
As introduced in the previous chapter, a vertical posture of the end-effector causes a particular task redundancy described by parameter ρ_4 , which represents the absolute orientation of the arm. Parameter ρ_4 can be arbitrarily set for any vertical configuration of the end-effector and it is not subject to any constraint. In fact, even if the last joint of the youBot arm presents physical limits, any end-effector configuration has an equivalent one rotated of 180 degrees around its principal axis, and so choosing between these two possible configuration allows to easily overcome the problem of the physical limits that affects the last joint of the arm.

A typical objective of redundancy resolution is to minimize the overall robot motion. In case of mobile manipulators, for example, it is a good practice to reduce the movements of the mobile platform with the aim of minimizing the error of the odometry system used for the localization of the robot. The task redundancy expressed by parameter ρ_4 can be used to achieve this behavior: indeed if the arm absolute orientation is kept aligned with the goal position during the approaching phase (Figure 3.3), the motion of the platform will be minimized. For this purpose the redundancy resolution can be solved imposing

$$\rho_4 = Arg(\boldsymbol{P_g} - \boldsymbol{P_a}) \tag{3.14}$$

where $\boldsymbol{P_g}$ is the position of the goal pose and $\boldsymbol{P_a}$ is the current position of the base of the arm. Expanding the previous equation, the following result can be found:

$$\rho_4 = atan2(y_g - (Y_b + x_d \sin(\theta_b)), x_g - (X_b + x_d \cos(\theta_b))) \tag{3.15}$$



P_b Mobile base position P_a Arm base position P_g Goal position

Figure 3.3 **Redundancy resolution for parameter ρ_4 .**

3.4 Arm-base displacement redundancy

The last redundancy that has to be analyzed is the rotational displacement between the base and the arm described by redundancy parameter ρ_1 . As shown in Equation (2.11), this redundancy parameter is equal to the value of the first joint of the manipulator, therefore it has to be subjected to the physical limits of that joint for preserving the consistency of the robot primary task. As discussed before, the problem of satisfying the joint physical limits is not particularly hard, and so a resolution strategy to better exploit the redundancy of parameter ρ_1 has to be found.

The first joint of the manipulator does not affect the manipulability of the whole system; intuitively the first joint contributes only to lateral movements of the and effector, which are already accomplished by the omnidirectional mobile base, thus the manipulability measure can not be consider for the redundancy resolution of parameter ρ_1 .

An obstacle avoidance behavior, that allows the base and the arm to cooperate with the purpose of assuming collision free configuration in a cluttered environment, can be a good strategy to exploit this kind of redundancy for mobile manipulator (Figure 3.4). For the particular case of KUKA youBot, due to the small size of its robotics arm, this behavior can be applied only to very particular situations, and thus it has not been pursued in this work.

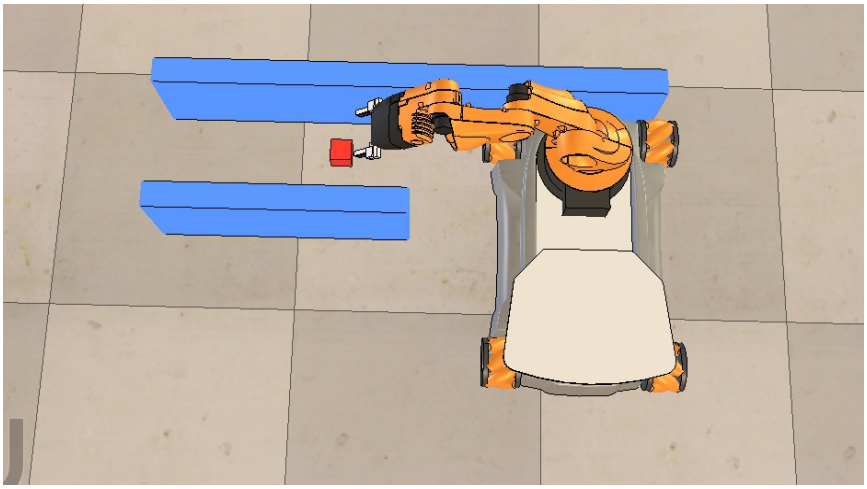


Figure 3.4 Redundancy resolution of parameter ρ_1 used in cluttered environment.

The arm-base displacement may also be used for navigation purposes. Navigation consists in the fundamental task for a mobile robot to localize itself and move in the environment from one position to another one. In this perspective a sensory system to perceive the environment is necessary to correctly retrieve information and plan the desired motion. Nowadays it is very popular to use depth cameras, devices that provide RGB images as well as accurate depth scansion, as sensory device to reconstruct the structure of the environment. The integration of such vision devices in the robot control system can be even more efficient if coupled with a dedicated redundancy resolution strategy. For example, in the particular case of KUKA youBot two depth camera devices can be adopted, one fixed on the mobile base and the other one mounted on the manipulator end-effector. The camera fixed on the platform has the purpose of tracking a certain goal object, that should be grasped and manipulated, while the other one has the ability to freely move and scan the environment as if it was a radar (Figure 3.5). The redundancy induced by parameter ρ_1 can then be used to keep the goal in the field of view of the fixed camera, while free motion of the other camera is still granted.

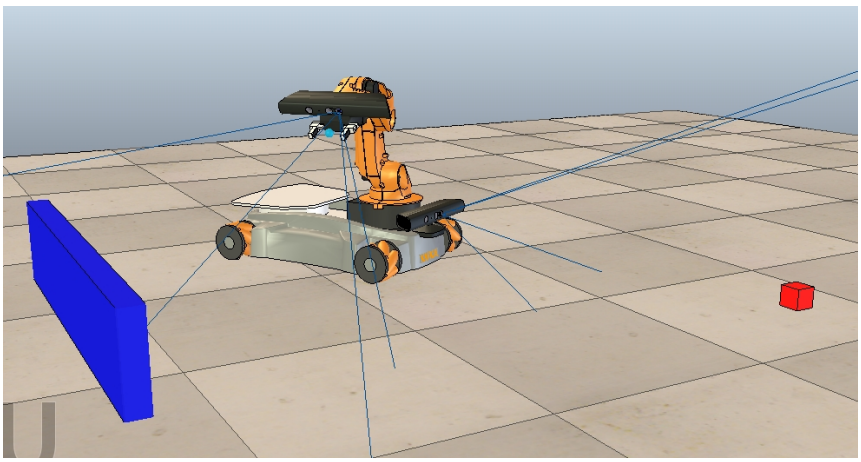


Figure 3.5 Redundancy resolution of parameter ρ_1 used in visual sensory system.

If the goal object has coordinates x_g and y_g, then the constraint to keep the goal in the field of view of the camera can be formulated as:

$$\left|\theta_b - atan2(y_g - Y_b, x_g - X_b)\right| < \frac{A}{2} \tag{3.16}$$

where A is the angular field of view of the camera. Despite this constraint the camera mounted on the end-effector preserve the ability to freely scan the arbitrary direction $\hat{\theta}$, setting the value of redundancy parameter ρ_1 as:

$$\rho_1 = \hat{\theta} - \theta_b \tag{3.17}$$

To give a brief application example, this technique could be used to develop a particularly efficient SLAM system [22], that has the ability to acquire sensor observation in the direction that minimize the error on the mapping and localization measures. Another application based on this particular redundancy resolution strategy may be the development of a visual odometry system, that should assist the standard one based on the wheel actuation in reconstructing the mobile base position.

Chapter 4

Navigation and obstacle avoidance

Navigation is a fundamental activity for mobile robots, that should be able to safely move from one position of the environment to another one, while avoiding collisions with the objects possibly disposed in it. Navigation has several objectives, as mapping the environment through some sensory system to extract a useful representation of it, localizing the robot in the perceived map and planning the necessary motion to reach the desired goal position. From a computational point of view, reconstructing an exact representation of the environment is a very complex task, as every possible configuration of the robot in the environment must be taken into account to perform an accurate motion planning. For this reason many navigation techniques [23] have been proposed over the years, based on the assumption that a discretized way to represent the environment is necessary to solve the motion planning problem. For example, in the *Cell decomposition* approach the environment is decomposed in a discrete grid of cells, then a value to every cell is assigned to determine if it is navigable or contains some obstacles. At the end a search algorithm like *Dijkstra* [24] or *A** [25] is applied to retrieve a valid path to reach the goal in the grid. The *Sampled based* methods randomly choose some feasible robot configurations and then try to connect them by finding pairs of those configurations that can be easily reached one from the other. In this way a navigation roadmap can be constructed and queried to find obstacle free paths. In the *Potential field* approaches, the robot motion is influenced by virtual forces, that have an attractive nature toward the goal position and a repulsive one from the obstacles. These virtual forces drive the robot toward the goal keeping it away from the obstacles in a reactive way.
In this chapter, a navigation technique based on potential fields and the necessary control strategy will be presented, focusing in particular on omnidirectional mobile robots, as the KUKA youBot.

4.1 Control of omnidirectional mobile platforms

Omnidirectional mobile robots are designed for 2D planar motions and are capable of translations and rotations, without the non-holonomic constraints typical of traditional wheeled robot. In a general form, the velocity control signal for an omnidirectional mobile platform can be expressed as:

$$\boldsymbol{u_b}=[v_x\ v_y\ \omega]^T \tag{4.1}$$

where v_x and v_y are the translational velocity component and ω is the angular velocity of the control signal. The vector $\boldsymbol{u_b}$ corresponds to the mobile robot velocity, expressed in the robot relative reference frame, that can be controlled in each time instant to obtain the desired motion. If $\hat{\boldsymbol{q}}_{\boldsymbol{b}}=[\hat{X}_b\hat{Y}_{b_d}\hat{\theta}_{b_d}]^T$ is the desired state of the platform expressed as absolute position and orientation, and $\boldsymbol{q_b}=[X_b\ Y_b\ \theta_b]^T$ is the actual state of the robot, then the absolute goal velocity $\boldsymbol{v_g}$ can be defined as:

$$\boldsymbol{v_g}=\dot{\hat{\boldsymbol{q}}}_{\boldsymbol{b}}+K_p(\hat{\boldsymbol{q}}_{\boldsymbol{b}}-\boldsymbol{q_b}) \tag{4.2}$$

where $\dot{\hat{\boldsymbol{q}}}_{\boldsymbol{b}}$ is the first time derivative of the desired state $\hat{\boldsymbol{q}}_{\boldsymbol{b}}$, and the diagonal matrix K_p represents the position gain. Therefore, to reach the desired location under the hypothesis of an empty environment, the velocity control signal can be expressed as

$$\boldsymbol{u_b}=R(\boldsymbol{q_b})\cdot\boldsymbol{v_g} \tag{4.3}$$

where $R(\boldsymbol{q_b})$ is a rotational matrix that maps the absolute velocities to the robot relative frame. However, a common objective for any navigation strategy is to achieve an obstacle avoidance behavior based on the environment perceived model. The environment model, in particular, can be constructed using common proximity sensors or more complex devices like laser scanners or depth cameras. In a complete general form the data acquired by the sensor system are collected in S_O , a set composed of the perceived positions $\boldsymbol{P_i}$, representing the absolute positions of the objects external

surfaces in the environment. Therefore, the control action for the omnidirectional mobile platform has been formulated as:

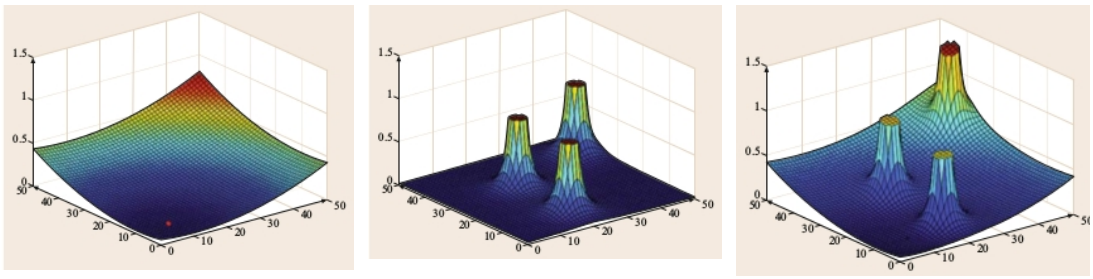
$$\boldsymbol{u_b} = R(\boldsymbol{q_b}) \cdot \boldsymbol{A_{oa}}(\boldsymbol{v_g}, S_O) \tag{4.4}$$

where $\boldsymbol{A_{oa}}(\boldsymbol{v_g}, S_O)$ is the obstacle avoidance algorithm which modifies the goal velocity $\boldsymbol{v_g}$ in accordance with the sensor perceptions stored in S_O so to avoid collisions.

4.2 Obstacle avoidance based on potential field

In 1986, Khatib proposed a technique for motion planning based on potential fields [26]; it does not explicitly construct a roadmap, but instead defines a differentiable real-valued function $U : \mathbb{R}^m \rightarrow \mathbb{R}$, called a potential function, that guides the motion of the moving robot. The potential is typically the combination of an attractive component $U_{att}(\boldsymbol{q_b})$, which pulls the robot toward the goal, and a repulsive component $U_{rep}(\boldsymbol{q_b})$, which pushes the robot away from the obstacles, as shown in Figure 4.1 . The gradient of the potential function defines a virtual force $\boldsymbol{F} = -\nabla U(\boldsymbol{q_b})$ that should be applied on the robot to achieve the desired motion. In this work, the virtual force has been interpreted as the velocity vector that should be applied on the robot to achieve the desired motion. Therefore, the velocity control signal can be calculated in a general form as:

$$\boldsymbol{u_b} = R(\boldsymbol{q_b})\, \boldsymbol{F} \quad . \tag{4.5}$$



Attractive function Repulsive component Potential function.

Figure 4.1 An attractive and repulsive component define a potential function.

Image taken from B. Siciliano, O. Khatib: Handbook of robotics - Chapter 5 Motion Planning, Springer, 2008.

However this technique does not always guarantee the robot to reach the desired position, because the gradient descent method can reach only a local minimum of the potential function, causing the robot to be stuck in some intermediate position (Figure 4.2).

Figure 4.2 **Common examples of potential field local minimum.**

Image taken from B. Siciliano, O. Khatib: Handbook of robotics - Chapter 5 Motion Planning, Springer, 2008.

In this work a *Rotational vector field [27]* has been used to construct a local minimum free potential function. The rotational field has the purpose of guiding the robot around the obstacles instead of simply being repulsed by them. In this way a correct navigation based on potential fields is possible even in complex environment. Precisely, if the distance between the perceived obstacle and the robot is below a certain threshold d_{rep} , the standard repulsive potential field is applied to avoid a possible collision. When instead the distance is between the thresholds d_{rep} and d_{rot} , the rotational field is applied to circumnavigate the obstacle.

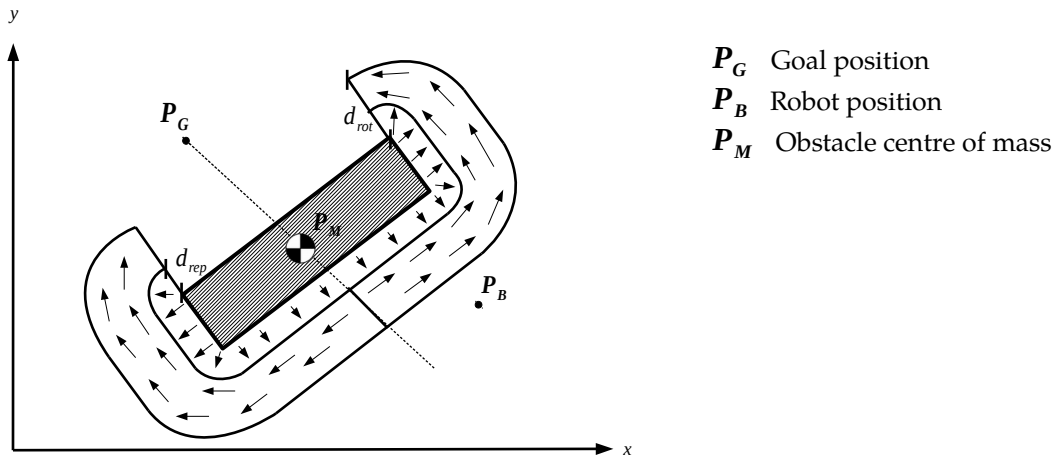


Figure 4.3 **Integration of repulsive and rotational potential fields.**

The attractive virtual force has been designed to be equal to the goal velocity:

$$\boldsymbol{F}_{att}(\boldsymbol{q}_b, \boldsymbol{q}_{b_d}) = \boldsymbol{v}_g \quad . \tag{4.6}$$

Given an obstacle perception $\boldsymbol{P}_O$ and the actual position of the robot $\boldsymbol{P}_B$, the repulsive virtual force has been designed to be proportional to the goal velocity with outgoing direction from the obstacle:

$$\boldsymbol{F}_{rep}(\boldsymbol{q}_b, \boldsymbol{P}_O, \boldsymbol{P}_G) = -\frac{\boldsymbol{P}_O - \boldsymbol{P}_B}{\|\boldsymbol{P}_O - \boldsymbol{P}_B\|}\|\boldsymbol{v}_g\| \quad . \tag{4.7}$$

The same design criteria has been utilized for the rotational virtual force, which is proportional to the goal velocity and has a direction that spins around the obstacle:

$$\boldsymbol{F}_{rot}(\boldsymbol{q}_b, \boldsymbol{P}_O, \boldsymbol{P}_G) = S_R \begin{bmatrix} -(Y_B - y_{P_O}) \\ X_B - x_{P_O} \end{bmatrix} \frac{\|\boldsymbol{v}_g\|}{\|\boldsymbol{P}_O - \boldsymbol{P}_B\|} \quad . \tag{4.8}$$

The parameter S_R defines the force direction of rotation and has value $+1$ for counter-clockwise rotations and -1 for clockwise rotations. The direction of rotation of the rotational field has a critical influence on the outcome of the navigation strategy. Indeed it is not possible to decide a fixed direction of rotation, because this could lead to the formation of local minimum points in the potential function. The direction of rotation must be decided in a specific way for every environment configuration (Figure 4.4) . In this work the direction of rotation is decided according to the position of the robot with respect to the obstacles and the goal position. The line joining the center of mass of the obstacle with the goal position divides the environment in two areas. Depending on the fact that the robot belongs to one or the other area, the direction of rotation is defined. In the following figure, on the left it is presented the case of fixed direction of rotation, which causes the formation of a local minimum point in the potential function, where the virtual forces

have opposite directions. On the right it is illustrated the correct situation with variable direction of rotation.

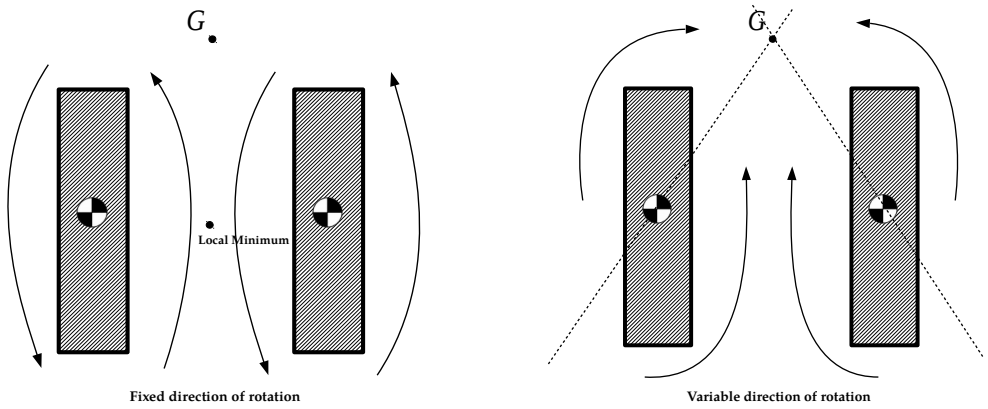


Figure 4.4 **Fixed and variable directions of rotation.**

The direction of rotation and the corresponding distinction between the two zones can by analytically obtained by the signed angle φ , defined by the intersection of segments $\overline{GM}$, which connects the goal position with obstacle center of mass, and $\overline{GB}$, which connects the goal position with the robot base (Figure 4.5).

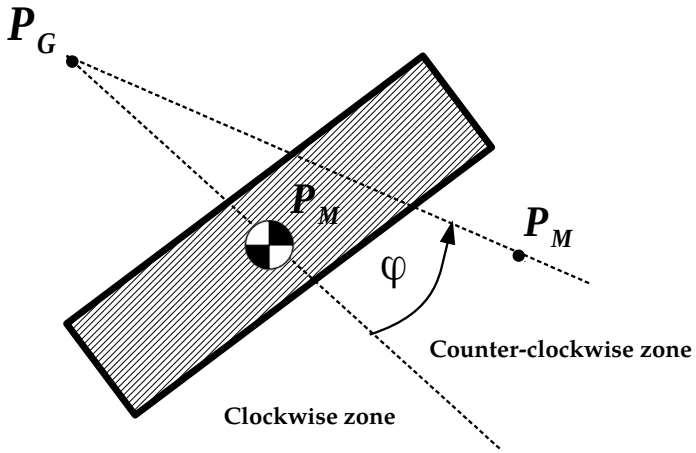


Figure 4.5 **Clockwise and counter-clockwise zones.**

The signed angle φ can be derived from the scalar and vector product of the two segments as

$$\varphi = atan2\left(\left\|(\boldsymbol{P_G}-\boldsymbol{P_M})\times(\boldsymbol{P_B}-\boldsymbol{P_M})\right\|,(\boldsymbol{P_G}-\boldsymbol{P_M})\cdot(\boldsymbol{P_B}-\boldsymbol{P_M})\right) \tag{4.9}$$

while the direction of rotation is equal to the opposite of the sign of the angle:

$$S_R = -sign(\varphi) \quad . \tag{4.10}$$

The reader should note that for a correct implementation of the rotation field it is necessary to define for every object in the environment its center of mass, starting from the perceived positions $\boldsymbol{P_i}$ collected in S_O . For this reason, a clustering technique has been introduced to subdivide the sensor perceptions $\boldsymbol{P_i}$ of S_O in some partitions C_j , so that every partition represents a different obstacle of the environment (See Figure 4.6). Given the maximum encumbrance of the robot d_{max} , every cluster C_j is defined by the relation:

$$C_j = \{\boldsymbol{P_i} \in S_0, \exists \boldsymbol{P_k} \in C_j \wedge \boldsymbol{P_k} \neq \boldsymbol{P_i}, \|\boldsymbol{P_i}-\boldsymbol{P_k}\| < d_{max}\} \quad . \tag{4.11}$$

which asserts that, if the distance between two sensor perceptions is less then d_{max} , these two perception belong to the same cluster C_j . Instead if there are no perceptions, whose distance from $\boldsymbol{P_i}$ is less then d_{max} , the perception $\boldsymbol{P_i}$ is the only one belonging to cluster C_i .

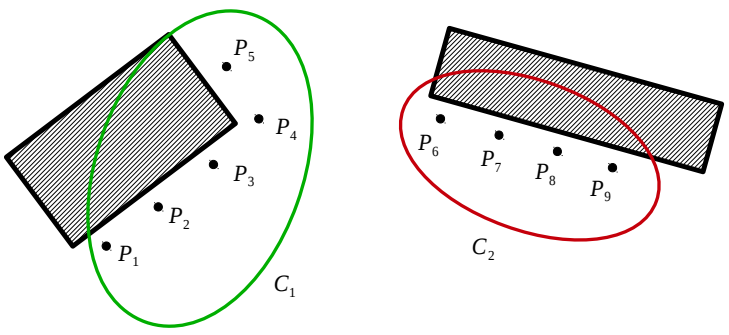


Figure 4.6 **Clustering on the sensor perceptions.**

As illustrated in Figure 4.7, the maximum encumbrance d_{max} corresponds to the maximum linear extension occupied by the robot mobile platform. In fact, if two obstacle perceptions dist less then d_{max} , it means that the robot may not be able to move between them, and so it is necessary to assign them to the same cluster, which will represent a certain obstacle.

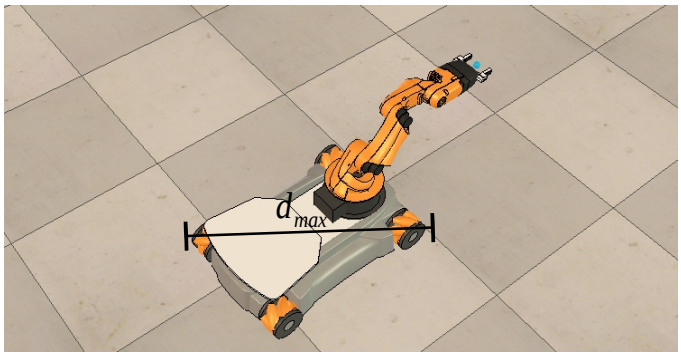


Figure 4.7 **KUKA youBot maximum encumbrance.**

The composition of the repulsive and rotational forces generated by the obstacle perception $\boldsymbol{P_O}$ has been obtained by the relation

$$\boldsymbol{F}_{\boldsymbol{rep/rot}}(\boldsymbol{q_b}, \boldsymbol{P_O}, \boldsymbol{P_G}) = \begin{cases} |\cos(\alpha)| \boldsymbol{F}_{rep}(\boldsymbol{q_b}, \boldsymbol{P_O}, \boldsymbol{P_G}) & if \ \|\boldsymbol{P_O} - \boldsymbol{P_G}\| < d_{rep} \\ \boldsymbol{F}_{rot}(\boldsymbol{q_b}, \boldsymbol{P_O}, \boldsymbol{P_G}) & if \ d_{rep} < \|\boldsymbol{P_O} - \boldsymbol{P_G}\| < d_{rep} \\ 0 & otherwise \end{cases} \quad (4.12)$$

where α is the angle between $\boldsymbol{v}_g$ and $\boldsymbol{F}_{rep}$ and can be computed as

$$\alpha = \frac{\|\boldsymbol{v}_g \cdot \boldsymbol{F}_{rep}\|}{\|\boldsymbol{v}_g\| \|\boldsymbol{F}_{rep}\|} \quad . \quad (4.13)$$

The term $|\cos(\alpha)|$ penalizes the repulsive force in case of a transversal motion of the robot with respect to the obstacle. Indeed if the robot is moving next to an obstacle and not straight toward it, it is not necessary for the repulsive virtual force to be excessively intense (see Figure 4.7).

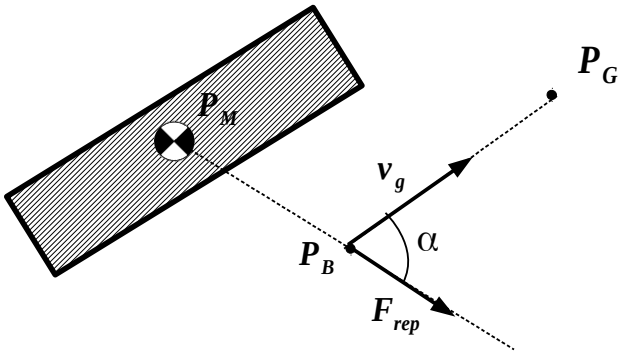


Figure 4.8 **Penalization of the repulsive virtual force.**

The overall virtual force to be applied on the robot considering all the sensor perceptions of S_O can be obtained by the weighted average of all the virtual forces $\boldsymbol{F}_{rep/rot}$:

$$\boldsymbol{F} = \boldsymbol{A}_{oa}(\boldsymbol{v}_g, S_O) = \boldsymbol{F}_{att}(\boldsymbol{q}_b) + \frac{\sum_{i=1}^{|S_0|} \frac{1}{d_i} \boldsymbol{F}_{rep/rot}(\boldsymbol{q}_b, \boldsymbol{P}_i)}{\sum_{i=1}^{|S_0|} \frac{1}{d_i}} \tag{4.14}$$

where d_i is the distance of perception $\boldsymbol{P}_i$ from the actual robot position.

An expedient to perform this kind of navigation in a dynamic environment with moving obstacles is to avoid to permanently store the sensor perceptions in S_O , but instead toassign to each position $\boldsymbol{P}_i$ a time interval, after which the observation is removed. In this way the set of sensor perception will contain only the last updated observations and will be able to deal with changes in the environment.

Chapter 5

Grasp synthesis

The main objective of robotic manipulation is to move objects in the workspace changing their position and orientation. To perform a manipulation task, a robot establishes physical contact with objects in the environment (typically through its gripper) and subsequently moves these objects by exerting forces and moments. One of the most critical aspects of manipulation is to decide how to grasp an object ensuring a firm contact between the robot gripper and the object surface. The process of deciding the correct way to grasp an object will be addressed in this work as *Grasp synthesis*. In this chapter an overview of existing grasping techniques will be presented, as well as a new method to perform grasp synthesis in a completely unknown environment, where shape, dimension and position of the objects to be manipulated is unknown to the robot. Particular attention is given to implementation issues of this grasping technique on mobile manipulators such as KUKA youBot.

5.1 Related works

Over the years many grasping strategies have been proposed to cope with different issues such as complexity of the object to be manipulated, structure and articulation of the gripper mechanism, complete or partial knowledge of the environment and type of available sensor devices. The different approaches can be subdivided into three different categories: *Model based*, *Recognition based* and *On-line based* grasp synthesis.

5.1.1 Model based grasp synthesis

The grasping strategies of this category are based on the assumption that a complete and detailed model of the object to be manipulated, as for example a CAD 3D model, is available. The a priori knowledge of the object model allows to perform very precise geometric and dynamic analysis and find

optimal grasp poses. A popular tool available is "*GraspIt!" [28][29]*, a very complete simulator specialized in grasp synthesis and analysis. The synthesis process of "GraspIt!" proceeds decomposing the model of the object in its primitive shapes, so that the approach directions, vectors perpendicular to the surfaces of the object, can be easily calculated. Then the gripper is positioned along these approach directions and the closure of its fingers around the object is simulated. To define the grasp pose, an optimized algorithm for collision and contact detection is used in order to determine the contact points between object and gripper. The simulator also offers the possibility to analise the grasp poses to check if the force and form closure properties [30] are satisfied. More in detail the analysis method implemented in "GraspIt!" searches for the maximum force disturbance applicable on the object such that the object does not change the position imposed by the gripper clutch [31]. The technique adopted by "GraspIt!" is often applied in case of articulated grippers with many degrees of freedom, as the grasping pose is defined starting from the gripper geometry, which is adapted to the object shape. Instead, in case of more simple grippers (as the parallel jaws mechanism with only one degree of freedom for opening and closing the two fingers), it is more convenient to construct the grasping pose starting from the object shape, as discussed in [32], where the concepts of *local* and *global accessibility* are introduced. Through local accessibility it is possible to identify on the object model pairs of faces, that can be utilized as grasping contact points. Global accessibility is used to select the grasping poses that can be realized in accordance with the robot kinematics and the obstacles located in the workspace.

The model based grasp synthesis is commonly adopted in industrial robotics, as the assumption of a complete knowledge of the environment and the possibility to have the detailed object models is consistent with the current industrial practices. Obviously this technique can not be applied to the case of an unknown environment, where the integration of a sensor system is indispensable to reconstruct the model of the objects that have to be grasped.

5.1.2 Recognition based grasp synthesis

The recognition based strategies store some object models and the corresponding grasping poses, precomputed with one of the model based

approaches, in a knowledge base constructed with a semi-automated process. In this way the grasp synthesis can be set as to a recognition problem, where an object in the environment should be put in relation with one of the models stored in the knowledge base, starting from some visual perception of the real object. In [33] the authors, after having realized a database of object models and grasp poses using "GraspIt!", perform the object recognition through the *Scale Invariant Feature Transform* algorithm. Furthermore the database can be manually updated with semantic informations, as for example points of the objects that are preferred as grasping regions or points that should not be touched. In [34] a similar approach is presented, where a *Nearest Neighbour Algorithm* is used for the object recognition, and the knowledge base is automatically updated after each grasping execution to keep track of the grasping poses that result more stable. Recognition based techniques are the most common ones, because they require a quite modest implementation effort, as some knowledge bases of object models and many recognition framework are freely available. These techniques, however, have evident limitations as they cannot plan grasping poses for objects not stored in the knowledge base, or in the worst case they could perform a wrong recognition, obtaining grasping poses that will fail at execution time.

5.1.3 On-line based grasp synthesis

On-line based techniques do not require the knowledge of the object models or a set of precomputed grasp poses, as the object models are reconstructed in real time only considering the sensors visual perception and the grasp synthesis is executed on the fly. The reconstruction of the object models is a critical phase of this methodology, because the information acquired through visual sensors is raw, partial and disturbed by noise. Hence designing efficient heuristics criteria is necessary to extract a semantic information from the raw data. In [35] *SFM – Structure From Motion* is used to acquire the raw data, which are elaborated with a *Voxeling* technique to reconstruct the surfaces and the volumes of the objects. From the generated surfaces, the grasping poses are calculated disposing the fingers of the gripper in a parallel direction to the object surfaces. Then some criteria to select the best grasping pose are introduced: the extension of the contact area between gripper and object, momentum balance, manipulability and robot motions. In [36] the authors

perform the model reconstruction with the *Superquadrics* approximation and then "GraspIt!" is used for the grasp poses generation and evaluation. In [37] a manipulator has been controlled with the purpose of empting a box full of objects. The models reconstruction is obtained through the *Height Accumulated Features*, which identifies the different object based on their height, and then a *Cell decomposition* algorithm selects the grasp pose that should be executed. In [38] the data acquisition is based on *Point Clouds, and* the model reconstruction and the grasp synthesis are performed in a continuous and simultaneous way. Moreover the robot motion is controlled in order to achieve an optimization on the grasp pose evaluation.

A particular methodology has been adopted in [39], where the authors integrate the grasp synthesis with a visual servoing technique. In this way is possible to guarantee a more robust grasp execution, that could cope with disturbances or with a dynamic environment, as in the case of moving objects. Another peculiar strategy is presented in [40], where the grasp synthesis is performed simultaneously with the motion planning phase. More in details the configurations obtained during the execution of the *RRT – Rapidly Random Trees* algorithm are also evaluated to verify if they are compatible with the geometry of the objects that have to be grasped.

In this work a new on-line technique for grasp synthesis has been developed. The complete synthesis process is composed of four different stages executed in sequence: Point Cloud Acquisition, Data Pre-processing, Grasp Generation, Grasp Selection. In the first phase, Point Cloud Acquisition, the outline of the target object is extracted from the image perceived through a visual sensor to acquire the model of the object as a point cloud. Then a pre-processing over the perceived data is necessary to remove the disturbance, to uniformly distribute the points on the object and to generate the surface normal vectors. During the grasp generation phase, the grasp poses compatible with both the geometry of the gripper and the object are calculated. Finally, the grasp pose that turns out to be more stable according to a heuristic evaluation is selected for execution. In this chapter the proposed grasp synthesis method will be discussed in details in the context of mobile manipulation, and in particular with reference to the KUKA youBot experimental platform.

5.2 Point Cloud Acquisition

An increasingly common practice in robotics is to adopt depth cameras, particular devices that work simultaneously as common RGB cameras and also as proximity sensors. These devices allow to acquire a so-called *Point Cloud*, a set of three-dimensional points that represent the external surface of the objects in the image frame. Point clouds are useful to acquire positions, shapes and dimensions of the object viewed by the camera and to reconstruct the state of the environment.

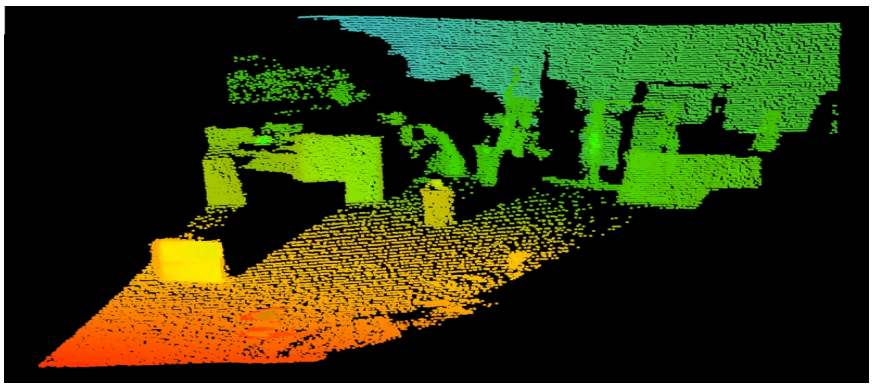


Figure 5.1 **Example of point cloud acquisition from a depth camera.**

A fundamental issue, that should be faced during the data acquisition, is how to identify the point cloud of a specific object and how to separate it from all the acquired points. For this purpose the built in RGB camera is used to acquire images, on which the classical computer vision techniques for features recognition are applied. In this work, objects are identified through their colour, hence the assumption is made that the colour of the objects that have be manipulated is known. This assumption on the objects colour is not at all restrictive, as the developed grasp synthesis process is based only on the acquired point cloud, and so any recognition or segmentation technique to identify the objects outline can be adopted instead of the colour-based one.
The data acquired from a depth camera can be expressed through the following matrices:

- I_{rgb} : a matrix of dimension $K \times J$ containing the RGB colour information for each pixel of the acquired image.

- I_d : a matrix of dimension $K \times J$ that assigns to each pixel its position in the environment, expressed in the camera reference frame.

Below a simple segmentation algorithm to retrieve the point cloud $\mathcal{P}$ of an object, given its colour C_{rgb} , will be presented. The points that do not belong to the target object are stored in the point cloud $\mathcal{P}_{obst}$, and they will be used in the grasp selection stage to reject grasp configurations that result in collision with other objects.

Data Acquisition Algorithm

Input:

I_{rgb}	- RGB image matrix.
I_d	- Depth image matrix.
T_c	- Homogeneous transformation from the absolute frame to the camera relative frame.
C_{rgb}	- Object colour.

Output:

$\mathcal{P}$	- Point Cloud of the target object expressed in absolute coordinates.
$\mathcal{P}_{obst}$	- Point Cloud containing all the points that do not belong to the target object expressed in absolute coordinates.

1. **For** $k = 1:1:K$
2. **For** $j = 1:1:J$
3. **If** $I_{rgb}(k, j)$ is close to C_{rgb}
4. $\boldsymbol{p} = T_c \cdot \boldsymbol{I_d}(k, j)$
5. **Add** $\boldsymbol{p}$ **to** $\mathcal{P}$
6. **else**
7. **Add** $\boldsymbol{p}$ **to** $\mathcal{P}_{obst}$
8. **end**
9. **end**
10. **end**

The color filtering, which has been introduced in the previous algorithm at step 3, can be realized by checking if the Red, Green and Blue values of a certain pixel are contained between an upper and lower threshold. The same color filtering strategy based on thresholds may be also applied on Hue, Saturation and Lightness values, which can by analytically derived from the RGB ones.

5.3 Data pre-processing

The point cloud, acquired from the depth camera and then outlined with the chosen segmentation technique, is still too raw and needs to be further elaborated before it can be used for the grasp generation stage. In particular, during the pre-processing stage, a density filter should be applied to remove some points that do not actually belong to the target object. The redundancy represented by too dense regions of the model is then removed through a uniformity filter, and finally the surface normal vectors are calculated for each point in the point of $\mathcal{P}$ to obtain a model richer of semantic information.

5.3.1 Density filter

Because of the uncertainty of the depth sensor, often some points of the point cloud, corresponding to the edges of the target object in the RGB image, do not actually belong to object model. For this reason, a density filter should be applied on the point cloud to remove regions with a low local density [41].
Let $\boldsymbol{p_i}$ be the *i*-th point of the point cloud $\mathcal{P}$ and $N_i(K)$ the *neighbour set*, defined as the set of K points closest to $\boldsymbol{p_i}$. Then the local density of point $\boldsymbol{p_i}$ is expressed as:

$$d_i(K)=\frac{1}{|N_i(K)|}\sum_{p_j\in N_i(K)}\|\boldsymbol{p_i}-\boldsymbol{p_j}\| \quad . \tag{5.1}$$

where the operator $\|.\|$ corresponds to the Euclidean norm of a vector.

Given the average point cloud local density

$$d_{AVG}(K)=\frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|} d_i(K) \tag{5.2}$$

and the corresponding standard deviation

$$d_{STD}(K)=\sqrt{\frac{\sum_{i=1}^{|\mathcal{P}|}(d_i(K)-d_{AVG}(K))^2}{|\mathcal{P}|}} \quad , \tag{5.3}$$

the density filter is defined as

$$F_d(\mathcal{P},K)=\{p_i\in\mathcal{P},\ d_i(K) > d_{AVG}(K)-3\mathrm{d}_{STD}(K)\} \quad . \tag{5.4}$$

The density filter has the effect of removing from the point cloud isolated points, e.g. those points whose local density is very low compared to the average of the acquired model.

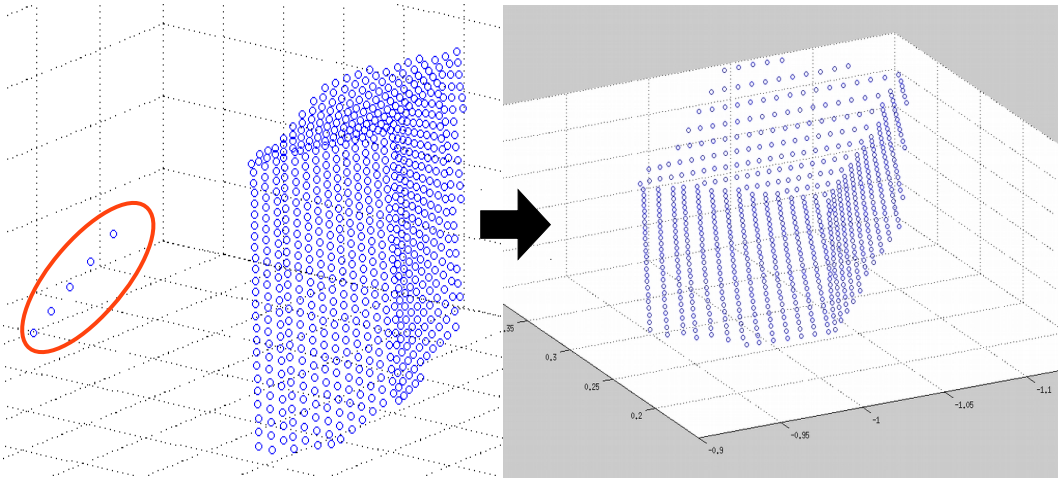


Figure 5.2 **Example of density filter application.**

5.3.2 Uniformity filter

In order to generate more precise surface normals and to remove the redundancy of information possibly included in the model, it is convenient to equally distribute the points on the surface of the point cloud. The uniformity filter indeed has the purpose of removing points that are too close to each other. Moreover this kind of filtering allows to obtain more efficient grasp selection end generation algorithm, as the redundancy of the model is removed and the overall points to be examined are few. The uniformity filter is defined as

$$F_u(\mathcal{P}, d_{min}) = \{ \boldsymbol{p}_i \in \mathcal{P}, \nexists \boldsymbol{p}_j \in \mathcal{P}, \|\boldsymbol{p}_j - \boldsymbol{p}_i\| < d_{min} \} \tag{5.5}$$

where d_{min} is the maximum admissible distance between two points.

5.3.3 Surface normals

For each point $\boldsymbol{p}_i$ of the point cloud $\mathcal{P}$ a normal vector to the surface of the object is calculated by finding the plane that best interpolates the points of neighbor set $N_i(K)$. Multiple linear regression is used as the basic mathematical tool to solve the plane estimation problem. Given the multiple linear regression model $y_k = \beta_0 + \beta_1 x_{1,k} + \beta_2 x_{2,k}$ with $x_{1,k} = x_{p_k}$, $x_{2,k} = y_{p_k}$ and $y_k = z_{p_k}$, using the well known *Least Square* resolution formula [42], the surface normal vector in $\boldsymbol{p}_i$ is obtained as:

$$\boldsymbol{n}_i = \frac{[\beta_1, \beta_2, 1]}{\sqrt{1 + \beta_1^2 + \beta_2^2}} \quad . \tag{5.6}$$

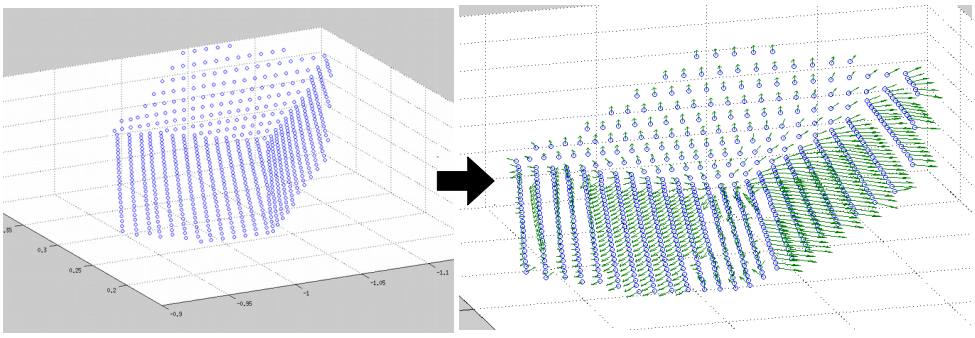


Figure 5.3 **Example of surface normals estimation.**

Furthermore it is possible to perform a smoothing operation on the surface normals to make them more regular where the surface of the object has a discontinuity, as for example near edges and corners. The smoothed surface normals can be calculated as:

$$\tilde{\boldsymbol{n}}_i(K) = \frac{1}{|N_i(K)|} \sum_{k=1}^{|N_i(K)|} \boldsymbol{n}_k \quad . \tag{5.7}$$

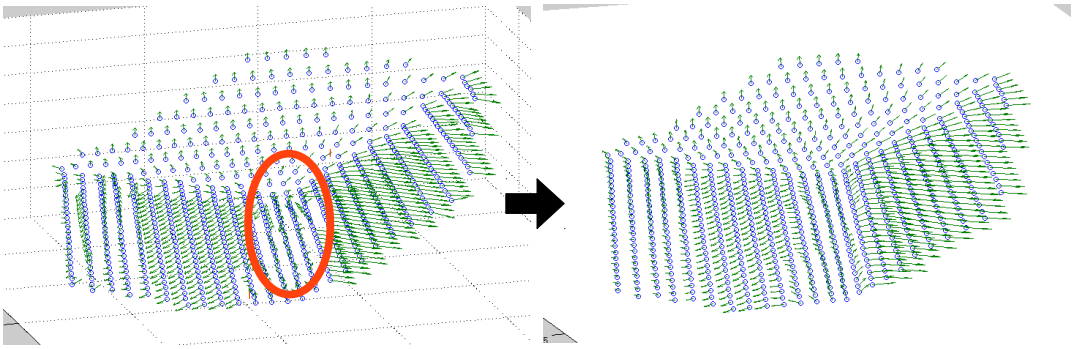


Figure 5.4 **Example of smoothing operation on the surface normals.**

5.4 Grasp generation

A grasping configuration should be derived considering the geometry of both the target object and the gripper itself. In this work only grippers with a single pair of parallel jaws, as the one mounted on KUKA youBot, are treated. The geometry of this kind of grippers can be described by four parameters L_1, L_2, L_3 and L_4, that define the sizes of the two jaws as illustrated in Figure 5.5 . In a general formulation a grasp pose G can be defined as

$$G = < \boldsymbol{p_G}, \boldsymbol{s_G}, \boldsymbol{a_G}, l_1 > \tag{5.8}$$

where $\boldsymbol{p_G}$ is the centered grasping position, $\boldsymbol{a_G}$ is the approach direction vector, $\boldsymbol{s_G}$ is the sliding direction vector and l_1 is the gripper opening dimension (see Figure 5.5).

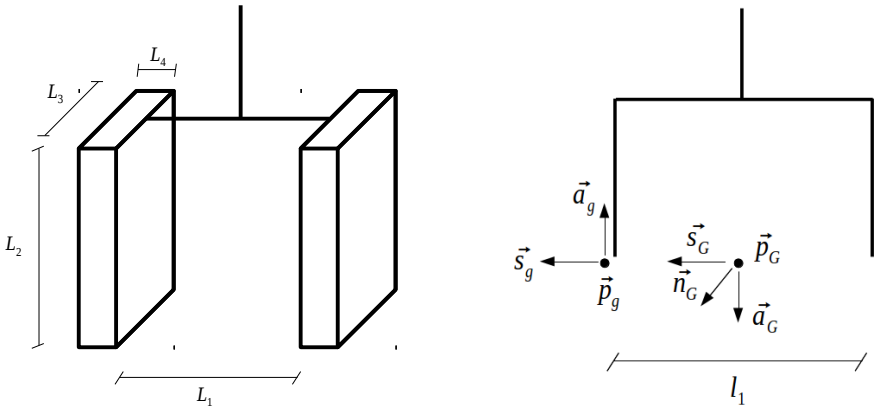


Figure 5.5 **Gripper and grasp pose parameters.**

In order to take into account the contact position between gripper and target object, in the grasp generation algorithm a more convenient notation of the grasp pose is adopted (see again Figure 5.5).:

$$\hat{G} = < \boldsymbol{p}_g, \boldsymbol{s}_g, \boldsymbol{a}_g, l_1 > \tag{5.9}$$

where $\boldsymbol{p}_g$ is the contact position of the left jaws, $\boldsymbol{s}_g$ is the sliding direction vector and $\boldsymbol{a}_g$ is the approach opposite direction vector. The two notation are equivalent and are related by the simple equations: $\boldsymbol{p}_G = \boldsymbol{p}_g - \frac{l_1}{2}\boldsymbol{s}_g$, $\boldsymbol{a}_G = -\boldsymbol{a}_g$, $\boldsymbol{s}_G = \boldsymbol{s}_g$.

The objective of the grasp generation stage is to find those gripper configurations that maintain its two fingers aligned with the object faces. Indeed, as proved in [43], keeping the gripper fingers parallel to the object faces guarantees for these simple kind of grippers the force closure property, namely the ability of a grasp configuration to resist any motion of the hold object, thanks to the contact forces that the gripper applies on it. As a point cloud is a simple collections of points, it has not an intrinsic definition of faces and surfaces, which should be reconstructed by the grasp generation algorithms. Two grasp generation algorithms have been developed in this work, the first one approximates the gripper structure with planes to find the contact points with the object, while the second one exactly reconstructs the object geometry. Both algorithms can generate grasping postures for objects with any shape, however the approximated one is very efficient and can be applied even to incomplete models of the objects, while the exact algorithm needs a complete and dense object model, but can generate very precise and stable grasp.

5.4.1 Locally approximated algorithm

This algorithm approximates the structure of the gripper with three plane arranged as in Figure 5.6 and searches for configuration of the gripper that keeps the parallelism between its fingers and faces of the object, while avoiding collisions with it in the neighborhood of the gripper structure.

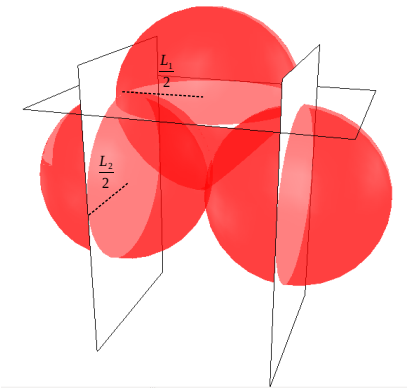


Figure 5.6 **Approximation of the gripper structure with planes.**

The function $CheckLocalPlane(\boldsymbol{p},\boldsymbol{n},\mathcal{P},d)$ has been introduced to verify if all the points of the point cloud $\mathcal{P}$ in the neighborhood of $\boldsymbol{p}$ with dimension d are behind the plane passing for $\boldsymbol{p}$ with norm $\vec{n}$.

$$CheckLocalPlane(\boldsymbol{p},\boldsymbol{n},\mathcal{P},d)=\begin{cases} true & if\ \forall\, \boldsymbol{p}_i\in\mathcal{P}, \|\boldsymbol{p}-\boldsymbol{p}_i\|<d \Rightarrow \boldsymbol{p}_i\boldsymbol{n}-\boldsymbol{p}\boldsymbol{n}<0 \\ false & otherwise \end{cases} \quad (5.10)$$

In order to completely explore all the possible grasp configurations, the algorithm is broken down into two phases. In the first phase the planes that approximate the gripper fingers are kept perpendicular to the norms $\boldsymbol{n}_i$ of each point $\boldsymbol{p}_i$ of the point cloud $\mathcal{P}$, while in the second phase configuration where the gripper palm is perpendicular to the surface normals are considered (see Figure 5.7).

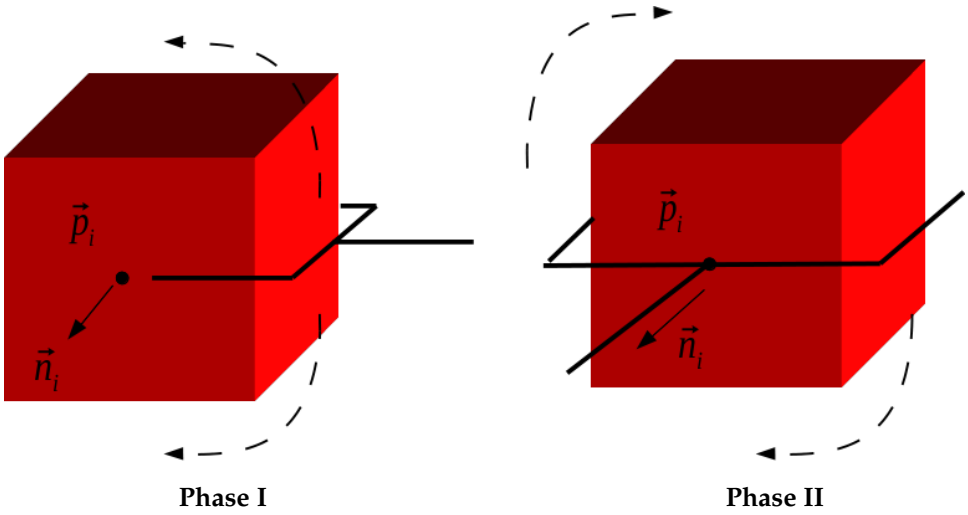


Figure 5.7 **Phases of the local approximated algorithm.**

The grasp generation algorithm pseudo-code is reported in the following table.

Locally approximated grasp generation algorithm

Input:
- Points $\boldsymbol{p_i}$ of point cloud $\mathcal{P}$ with the respective surface normals $\boldsymbol{n_i}$.
- Gripper parameters L_1, L_2 .

Output:
- List of all the admissible grasp pose $\hat{G}$.

1. **For each** $\boldsymbol{p_i} \in \mathcal{P}$ *(Phase I)*
2. $\quad \boldsymbol{p_g} = \boldsymbol{p_i}$
3. $\quad \boldsymbol{s_g} = \boldsymbol{n_i}$
4. $\quad$ **If** $\mathit{CheckLocalPlane}\left(\boldsymbol{p_g}, \boldsymbol{s_g}, \mathcal{P}, \frac{L_2}{2}\right)$
5. $\quad\quad$ **For each** $\boldsymbol{a_g} \perp \boldsymbol{s_g}$
6. $\quad\quad\quad$ **For** $l_1 = L_{min} : L_{step} : L_1$

7. **If** $CheckLocalPlane(\boldsymbol{p}_g - l_1 \boldsymbol{s}_g, -\boldsymbol{s}_g, \mathcal{P}, \frac{L_2}{2}) \wedge CheckLocalPlane(\boldsymbol{p}_g - \frac{l_1}{2}\boldsymbol{s}_g + L_2 \boldsymbol{a}_g, \boldsymbol{a}_g, \mathcal{P}, \frac{L_1}{2})$
8. **Add** $\hat{G} = <\boldsymbol{p}_g, \boldsymbol{s}_g, \boldsymbol{a}_g, l_1>$ **to output list.**
9. **End**
10. **End**
11. **End**
12. **End**
13. **End**
14. **For each** $\boldsymbol{p}_i \in \mathcal{P}$ *(Phase II)*
15. $\boldsymbol{a}_g = \boldsymbol{n}_i$
16. **If** $CheckLocalPlane(\boldsymbol{p}_i, \boldsymbol{a}_g, \mathcal{P}, \frac{L_1}{2})$
17. **For each** $\boldsymbol{s}_g \perp \boldsymbol{a}_g$
18. **For** $l_1 = L_{min} : L_{step} : L_1$
19. $\boldsymbol{p}_g = \boldsymbol{p}_i - L_2 \boldsymbol{a}_g + \frac{l_1}{2}\boldsymbol{s}_g$

    $\boldsymbol{p}_g' = \boldsymbol{p}_i - L_2 \boldsymbol{a}_g - \frac{l_1}{2}\boldsymbol{s}_g$
20. **If** $CheckLocalPlane(\boldsymbol{p}_g - \frac{L_2}{2}\boldsymbol{a}_g, \boldsymbol{s}_g, \mathcal{P}, \frac{L_2}{2}) \wedge CheckLocalPlane(\boldsymbol{p}_g' - \frac{L_2}{2}\boldsymbol{a}_g, -\boldsymbol{s}_g, \mathcal{P}, \frac{L_2}{2})$
21. **Add** $\hat{G} = <\boldsymbol{p}_g, \boldsymbol{s}_g, \boldsymbol{a}_g, l_1>$ **to output list**
22. **End**
23. **End**
24. **End**
25. **End**
26. **End**

5.4.2 Exact algorithm

The exact algorithm reconstructs the structure of the gripper with three cuboids, parallelepipedal regions of the space, as shown in Figure 5.8 . The grasp configurations are searched, as in the approximated algorithm, by keeping the fingers of the gripper parallel to the object faces, and discarding those configuration that result in collision with the points of the object point cloud.

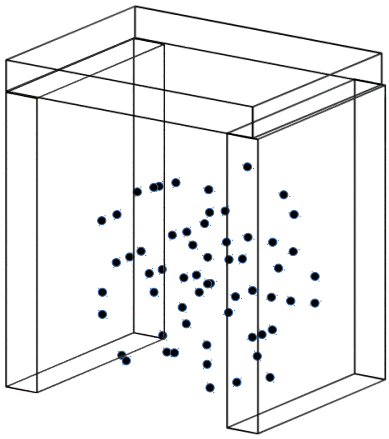
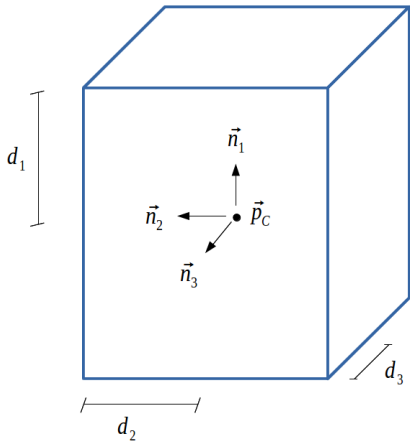


Figure 5.8 **Gripper structure reconstructed with cuboids.**

Let the cuboid $C=\langle \boldsymbol{p_C}, \boldsymbol{n_1}, d_1, \boldsymbol{n_2}, d_2, \boldsymbol{n_3}, d_3 \rangle$ be the portion of the space delimited by the six planes listed in the following table.



Plane	Point	Norm
#1	$\boldsymbol{p_C}+d_1\boldsymbol{n_1}$	$\boldsymbol{n_1}$
#2	$\boldsymbol{p_C}-d_1\boldsymbol{n_1}$	$-\boldsymbol{n_1}$
#3	$\boldsymbol{p_C}+d_2\boldsymbol{n_2}$	$\boldsymbol{n_2}$
#4	$\boldsymbol{p_C}-d_2\boldsymbol{n_2}$	$-\boldsymbol{n_2}$
#5	$\boldsymbol{p_C}+d_3\boldsymbol{n_3}$	$\boldsymbol{n_3}$
#6	$\boldsymbol{p_C}-d_3\boldsymbol{n_3}$	$-\boldsymbol{n_3}$

The function $CheckCuboid(C,\mathcal{P})$ checks if every point $\boldsymbol{p}_i$ of point cloud $\mathcal{P}$ is outside the cuboid C.

$$CheckCuboid(C,\mathcal{P})=\begin{cases} true & if\ \nexists\, \boldsymbol{p}_i\in\mathcal{P}, \boldsymbol{p}_i\in C \\ false & otherwise \end{cases} \tag{5.11}$$

Exact grasp generation algorithm

Input:
- Points $\boldsymbol{p}_i$ of point cloud $\mathcal{P}$ with the respective surface normals $\boldsymbol{n}_i$.
- Gripper parameters L_1, L_2, L_3, L_4.

Output:
- List of all the admissible grasp poses $\hat{G}$.

1. **For each** $\boldsymbol{p}_i\in\mathcal{P}$ *(Phase I)*
2. $\quad\boldsymbol{p}_g=\boldsymbol{p}_i$
3. $\quad\boldsymbol{s}_g=\boldsymbol{n}_i$
4. $\quad$**For each** $\boldsymbol{a}_g\perp\boldsymbol{s}_g$
5. $\quad\quad C_1=<\boldsymbol{p}_g+\frac{L_4}{2}\boldsymbol{s}_g, \boldsymbol{s}_g, \frac{L_4}{2}, \boldsymbol{a}_g, \frac{L_2}{2}, \boldsymbol{s}_g\times\boldsymbol{a}_g, \frac{L_3}{2}>$
6. $\quad\quad$**If** $CheckCuboid(C_1,\mathcal{P})$
7. $\quad\quad\quad$**For** $l_1=L_{min}:L_{step}:L_1$
8. $\quad\quad\quad\quad C_2=<\boldsymbol{p}_g-\frac{3L_4+2l_1}{2}\boldsymbol{s}_g, \boldsymbol{s}_g, \frac{L_4}{2}, \boldsymbol{a}_g, \frac{L_2}{2}, \boldsymbol{s}_g\times\boldsymbol{a}_g, \frac{L_3}{2}>$
9. $\quad\quad\quad\quad C_3=<\boldsymbol{p}_g+\frac{L_4+l_1}{2}\boldsymbol{s}_g+\frac{L_2+L_4}{2}\boldsymbol{a}_g, \boldsymbol{s}_g, \frac{L_4+l_1}{2}, \boldsymbol{a}_g, \frac{L_4}{2}, \boldsymbol{s}_g\times\boldsymbol{a}_g, \frac{L_3}{2}>$
10. $\quad\quad\quad\quad$**If** $CheckCuboid(C_2,\mathcal{P})\wedge CheckCuboid(C_3,\mathcal{P})$
11. $\quad\quad\quad\quad\quad$**Add** $\hat{G}=<\boldsymbol{p}_g-\frac{L_2}{2}\boldsymbol{a}_g, \boldsymbol{s}_g, \boldsymbol{a}_g, l_1>$ **to output list**
12. $\quad\quad\quad\quad$**End**
13. $\quad\quad\quad$**End**
14. $\quad\quad$**End**

15. **End**
16. **End**
17. **For each** $\boldsymbol{p}_i \in \mathcal{P}$ *(Phase II)*
18. $\boldsymbol{a}_g = \boldsymbol{n}_i$
19. **For each** $\boldsymbol{s}_g \perp \boldsymbol{a}_g$
20. **For** $l_1 = L_{min} : L_{step} : L_1$
21. $\boldsymbol{p}_g = \boldsymbol{p}_i + \frac{l_1 + 2L_4}{2}\boldsymbol{s}_g - \frac{L_2 + 2L_4}{2}\boldsymbol{a}_g$
22. $C_1 = < \boldsymbol{p}_g + \frac{L_4}{2}\boldsymbol{s}_g, \boldsymbol{s}_g, \frac{L_4}{2}, \boldsymbol{a}_g, \frac{L_2}{2}, \boldsymbol{s}_g \times \boldsymbol{a}_g, \frac{L_3}{2} >$
23. $C_2 = < \boldsymbol{p}_g - \frac{3L_4 + 2l_1}{2}\boldsymbol{s}_g, \boldsymbol{s}_g, \frac{L_4}{2}, \boldsymbol{a}_g, \frac{L_2}{2}, \boldsymbol{s}_g \times \boldsymbol{a}_g, \frac{L_3}{2} >$
24. $C_3 = < \boldsymbol{p}_g + \frac{L_4 + l_1}{2}\boldsymbol{s}_g + \frac{L_2 + L_4}{2}\boldsymbol{a}_g, \boldsymbol{s}_g, \frac{L_4 + l_1}{2}, \boldsymbol{a}_g, \frac{L_4}{2}, \boldsymbol{s}_g \times \boldsymbol{a}_g, \frac{L_3}{2} >$
25. **If** $CheckCuboid(C_1, \mathcal{P}) \wedge CheckCuboid(C_2, \mathcal{P}) \wedge CheckCuboid(C_3, \mathcal{P})$
26. **Add** $\hat{G} = < \boldsymbol{p}_g - \frac{L_2}{2}\boldsymbol{a}_g, \boldsymbol{s}_g, \boldsymbol{a}_g, l_1 >$ **to output list**
27. **End**
28. **End**
29. **End**
30. **End**

5.5 Grasp selection

Among all the generated grasp poses, it is necessary to select the one that best guarantees a successful grasping execution. To this end, some performance indexes has been proposed in [35][44] to evaluate each grasping configuration. Three different categories of metrics have been considered in this work: *Feasibility*, *Stability* and *Robot Motion*. The feasibility indexes are used to discard those grasp configurations that the robot kinematics can not accomplish or which result in collisions with other objects disposed in the

workspace. Stability indexes evaluate the grasp poses preferring those that guarantee a firm grasp, resistant to external disturbances or wrenches that can act on the object. Robot motions metrics prefer grasp postures that require less movement of the mobile platform for mobile manipulators, or joints displacement in case of standard manipulators. Finally a global performance index should be designed to include all the proposed metrics in a unique comprehensive measure. The grasp pose that obtains the best evaluation by the global performance index is the one chosen for execution. All the different designed metrics, that compose the global performance index, will be presented here in detail.

5.5.1 Feasibility

For each grasp pose G generated by the grasp synthesis algorithm it is necessary to check if it is contained in the reachable dexterous workspace of the robot. For the KUKA youBot robot, for example, the dexterous workspace condition expressed by Equation (2.38) should be used to verify whether the robot can reach the considered grasp position. In order to verify if a grasp pose with position $\boldsymbol{p_G}$ and approach direction $\boldsymbol{a_G}$ is reachable, the following relation can be derived from the dexterous workspace conditions:

$$(a_2+a_3)^2-\left(z_{\vec{p}_G}-z_d-d_1-d_5\sin\left(atan2\left(z_{\boldsymbol{a}_G},\sqrt{x_{\boldsymbol{a}_G}^2+y_{\boldsymbol{a}_G}^2}\right)\right)\right)^2>0 \quad . \qquad (5.12)$$

Furthermore, also grasp configurations that result in collision with other objects disposed in the environment should be discarded. To this end, the point cloud $\mathcal{P}_{obst}$, containing all the points acquired by the depth camera that do not belong to the target object, must be taken into account for a collision detection. For example each link of the robot and the structure of the base can be modeled with an adequate cuboid and then the *CheckCuboid* function itself can be used for collision detection.

Note that the feasibility indexes do not directly evaluate the grasping poses, but reject not achievable ones.

5.5.2 Stability

Without a complete dynamic model of the object, it is not possible to evaluate the stability of the generated grasp poses with the traditional force closure method. Hence some heuristics should be introduced to estimate how firm a grasp configuration is. For instance the distance of the grasping position from the centroid of the objects can be adopted to evaluate the overall gravity balance imposed by the gripper posture. Obviously the closer to the centroid the grasping is executed, the more it will result stable.

If $\boldsymbol{p}_{COM}=\frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|}\boldsymbol{p}_i$ is the centroid of point cloud $\mathcal{P}$, then the heuristic index to be used to evaluate the stability of the grasping pose G can be expressed as:

$$S_{COM}(G)=\frac{1}{\|\boldsymbol{p}_G-\boldsymbol{p}_{COM}\|} \quad . \tag{5.13}$$

Given that the perceived point cloud is often an incomplete model of the real object, and that some of its regions may be more dense then others, a better estimation of the object center of mass can be expressed as

$$\boldsymbol{p}_{ECOM}=\left[\frac{x_{max}+x_{min}}{2},\frac{y_{max}+y_{min}}{2},\frac{z_{max}+z_{min}}{2}\right]^T \tag{5.14}$$

where x_{max}, x_{min}, y_{max}, y_{min}, z_{max}, z_{min} are the maximum and minimum values of the Cartesian coordinates of all the points of the object point cloud in the world frame. The respective heuristic index can be formulated as

$$S_{ECOM}(G)=\frac{1}{\|\boldsymbol{p}_G-\boldsymbol{p}_{ECOM}\|} \quad . \tag{5.15}$$

Another heuristic to evaluate the stability of a certain grasp posture consists in estimating the extension of the contact area between the gripper fingers and the object faces. To this end let $C_{f1}=\langle \boldsymbol{p_G}+\frac{L_2}{2}\boldsymbol{a_G}, \boldsymbol{a_G}, \frac{L_2}{2}, \boldsymbol{s_G}, \frac{L_4}{2}+\epsilon, \boldsymbol{a_G}\times\boldsymbol{s_G}, \frac{L_3}{2}\rangle$ and $C_{f2}=\langle \boldsymbol{p_G}+\frac{L_2}{2}\boldsymbol{a_G}-l_1\boldsymbol{s_G}, \boldsymbol{a_G}, \frac{L_2}{2}, \boldsymbol{s_G}, \frac{L_4}{2}+\epsilon, \boldsymbol{a_G}\times\boldsymbol{s_G}, \frac{L_3}{2}\rangle$ be the cuboids which correspond to the fingers of the gripper slightly extended by the positive parameter ϵ in the sliding direction with respect to the considered grasp pose $G=<\boldsymbol{p_G}, \boldsymbol{s_G}, \boldsymbol{a_G}, l_1>$. The set of all the points contained in C_{f1} and C_{f2} is expressed as

$$P_{contact}(G,\mathcal{P})=\{\boldsymbol{p_i}\in\mathcal{P}, \boldsymbol{p_i}\in C_{f1}\vee \boldsymbol{p_i}\in C_{f2}\} \tag{5.16}$$

and then the respective heuristic index can be formulated as:

$$S_{CONTACT}(G,\mathcal{P})=|P_{contact}(G,\mathcal{P})| \quad . \tag{5.17}$$

5.5.3 Robot motion

Choosing a grasp configuration that requires the minimum movement of the robot is a necessary measure to optimize the energy consumption and reduce the execution time of the grasping task. Furthermore, in case of mobile manipulators, an odometry system is often used to keep track of the robot position. Hence minimizing the movement of the mobile base is a good strategy to reduce the odometry error, which increases over time during the robot motion. The robot configuration $\boldsymbol{q_G}$, needed to execute the grasping pose $G=<\boldsymbol{p_G}, \boldsymbol{s_G}, \boldsymbol{a_G}, l_1>$, can be computed using the closed form inverse kinematics algorithm presented in Chapter 2. In particular the following relations hold: $\boldsymbol{p_G}=[x_g\, y_g\, z_g]^T$, $\boldsymbol{Z_e}=\boldsymbol{a_G}$, $\boldsymbol{X_e}=\boldsymbol{a_G}\times\boldsymbol{s_G}$.

Given the present robot state $\boldsymbol{q_{now}}$ and the chosen grasp configuration $\boldsymbol{q_G}$, an index measure to evaluate the robot motions can be expressed as:

$$S_{RM}(G)=\frac{1}{\|\boldsymbol{q_{now}}-\boldsymbol{q_G}\|} \quad . \tag{5.18}$$

5.5.4 Global performance index

The global performance index, adopted during the experimental testings on KUKA youBot robot, to evaluate a grasping pose G for an object with point cloud $\mathcal{P}$ has been designed as

$$S(G,\mathcal{P}) = W(G) \cdot S_{CONTACT}(G,\mathcal{P}) \cdot S_{ECOM}(G) \tag{5.19}$$

where $W(G)$ is a penalization factor that prefers vertical configurations of the robot end-effector and it is defined as

$$W(G) = \begin{cases} 2 & \text{if } \boldsymbol{a}_G \parallel [0,0,1]^T \\ 1 & \text{otherwise} \end{cases} . \tag{5.20}$$

Indeed, in case of a KUKA youBot mobile manipulator, vertical configurations of the end-effector allow to take advantage of the particular task redundancy expressed by ρ_4 . As discussed in chapter 3, dedicated to redundancy resolution, this particular situation can be exploited so that the mobile base movements are minimized. Hence, vertical grasping configurations have been preferred exploiting the penalization factor $W(G)$.

Chapter 6

Pick and place operations with KUKA youBot

Pick and place operations are a typical example of robotic manipulation, where a certain object is rearranged in a desired location of the workspace. Mobile manipulators, with their capability to move around the environment, have no workspace limitations and can then be used for complex pick and place operations, both in service and industrial robotics. For example, they can be used to sort the objects disposed in a disordered room or to perform inventory movement in a warehouse. All the topics discussed in the previous chapters, like redundancy resolution, navigation and grasp synthesis, have been put together to develop a pick and place framework on a KUKA youBot mobile manipulator. The pick and place operations consist in localizing the object to be manipulated, picking it up and moving it in a desired position. Both the environment and the information regarding shape, position and dimension of the objects that should be manipulated are unknown to the robot. For this reason, a depth camera has been mounted on the robot platform and it is used to acquire the necessary visual information for navigation and grasp synthesis. The desired task is described in an abstract way, providing to the robot only the colour of the object that should be grasped, and the colour of the surface where the object has to be released. Thanks to the developed framework, the robot is able to autonomously explore the environment, reconstruct the model of the objects, plan and execute the grasping and placing operations.
In this chapter, the architecture of the developed system as well as some implementation issues will be presented.

6.1 System architecture

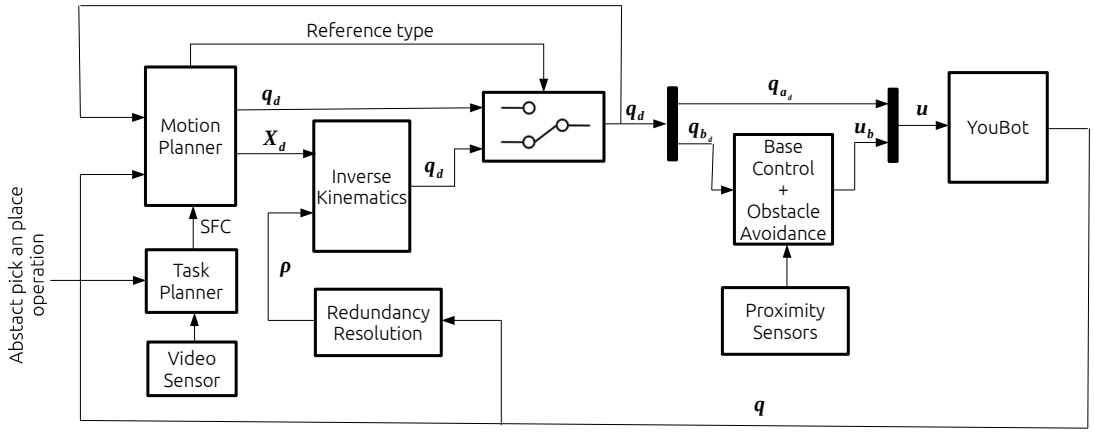
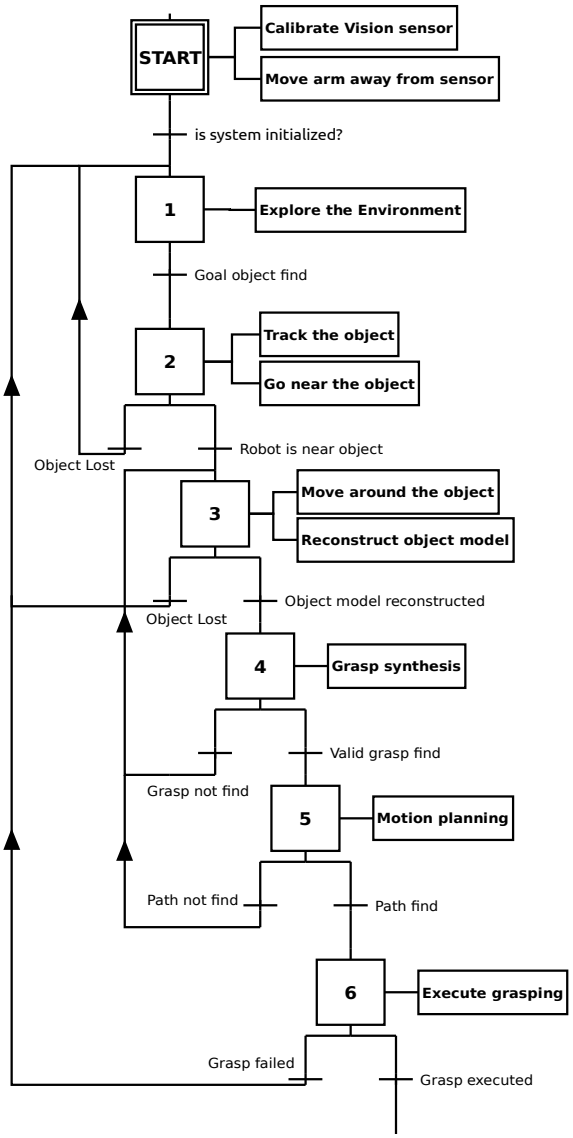


Figure 6.1 **System architecture.**

The architecture of the developed system is presented in the block diagram in Figure 6.1. The abstract description of the desired pick and place operation, containing the objects colour information, is provided to the Task Planner block, which has the purpose of generating a control structure in the *SFC – Sequential Function Chart* [45] formalism. This particular automaton, considering also the sensor visual perceptions, coordinates and supervises the overall system, decomposing the pick and place operation in all the sub-task necessary to perform the desired operation. Every sub-task is interpreted by the Motion planner, which generates the necessary trajectories in the Cartesian space or directly in the joint space. In case of Cartesian trajectories, the closed form inverse kinematics algorithm discussed in Chapter 2 is used to obtain the corresponding joint values. The Base Control, besides providing the velocity set points necessary for controlling the mobile platform, realizes the obstacle avoidance strategy based on rotational potential fields, in accordance with the data acquired by the proximity sensors, as discussed in Chapter 4. Finally the Redundancy Resolution block resolves the redundancy based on the robot actual state, optimizing the whole robot manipulability and minimizing the movements of the platform, as presented in Chapter 3.

6.2 Task Planner

The high-level control executed by the Task Planner has been designed following the Sequential Function Chart formalism. The complete diagram will be here presented in detail.



START. The manipulator is positioned in a configuration that does not obstruct the depth camera. The depth camera is calibrated to localize the requested colour.

STEP 1. The mobile platform executes a rotation around its axis so that the vision sensor can visualize the surrounding environment.

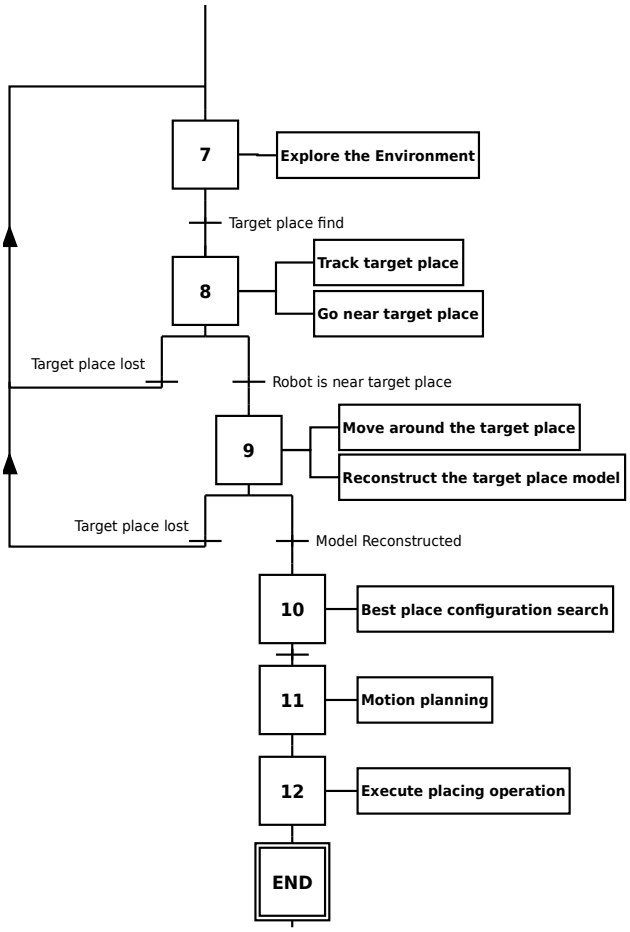
STEP 2. After localizing the object, the robot moves toward it. The camera continuously tracks the object so that the robot can follow it in case of movement.

STEP 3. The robot executes a rotation around the target object, allowing the depth camera to acquire its point cloud model from various angles.

STEP 4. The grasp synthesis is performed in order to decide how to grasp the object.

STEP 5. The necessary trajectories to execute the grasping task are planned.

STEP 6. The grasping task is executed.



STEP 7. The robot performs a rotation around its axis to explore the environment and localize the release surface.

STEP 8. The robot moves toward the target release surface, while the camera keeps tracking it.

STEP 9. The robot executes a rotation around the release surface, allowing the depth camera to reconstruct its point cloud model from various angles.

STEP 10. The best way to place the object on the surface, maximising the contact area between the objects, is calculated.

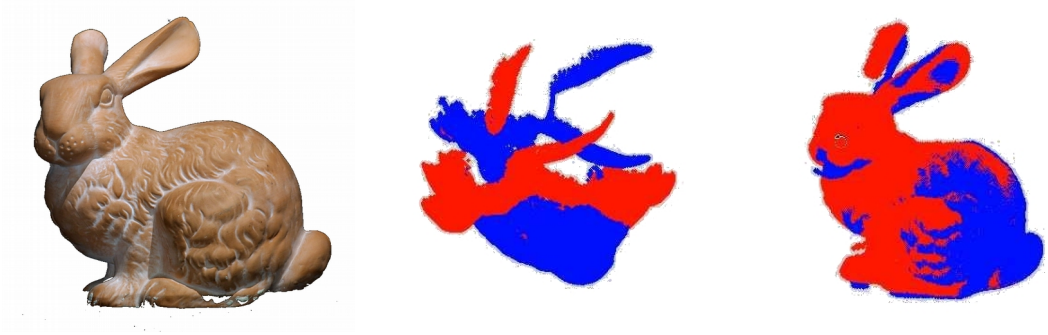
STEP 11. The necessary trajectories to execute the placing task are planned.

STEP 12. The placing task is executed.

6.3 Point cloud and grasping registration

The grasp synthesis process, described in Chapter 5, allows to determine a grasping configuration to pick up a certain object, starting from the data acquired by the depth camera device. Suppose that the point cloud of the object that has to be grasped has been acquired at time t_G and that the pick up task has been executed at time t_E, after the robot has performed the necessary base motion to reach the object. In the elapsed time between t_G and t_E, the robot localization is subject to the mechanical odometry based on the wheels actuation, causing the chosen grasp pose $G = <\boldsymbol{p_G}, \boldsymbol{s_G}, \boldsymbol{a_G}, l_1>$ to be affected by the same positional error of the odometry system.

Furthermore, the object could move during the motion of the mobile platform, leaving the chosen grasp pose completely inconsistent. In order to face this situations, during the time interval $\delta t = t_E - t_G$, the grasp pose should be updated according to the depth camera perceptions.
The process of finding the frame transformation between two point clouds observation of the same feature by different point of views is called *Point Cloud Registration* [46]. More precisely, given two point clouds $\mathcal{P}_A$ and $\mathcal{P}_B$ of the same feature, but acquired from different points of view, the point cloud registration technique allows to find the homogeneous transformation matrix T_B^A , that, applied on the points of $\mathcal{P}_B$, best matches the points of $\mathcal{P}_A$ (see Figure 6.2).



Bunny Object Unregistered point clouds Registered point clouds

Figure 6.2 Example of point cloud registration.

Iterative Closest Point – ICP [47] is one of the most used algorithms for point cloud registration. In this algorithm, one point cloud, the reference, is kept fixed, while the other one, the source, is transformed to best match the reference. The algorithm iteratively revises the transformation (combination of translation and rotation) needed to minimize the distance from the source to the reference point cloud. Essentially, the algorithm steps are:

1. For each point in the source point cloud, find the closest point in the reference point cloud.

2. Estimate the combination of rotation and translation using a mean square error cost function that will best align each source point to its match found in the previous step.
3. Transform the source points using the obtained transformation.
4. Iterate until the desired accuracy or the maximum number of iteration is reached.

To apply this registration technique to the grasping case, let $\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2 \ldots \mathcal{P}_n$ be the point clouds of the object that should be grasped, acquired respectively at time instants $t_0, t_1, t_2 \ldots t_n$. The Iterative Closest Point can be used to find the homogeneous transformations T_{i+1}^{i} that execute the best matching between two consecutive acquired point clouds :

$$\mathcal{P}_i \approx T_{i+1}^{i} \mathcal{P}_{i+1} \quad . \tag{6.1}$$

If the grasp pose $G = <\boldsymbol{p}_G, \boldsymbol{s}_G, \boldsymbol{a}_G, l_1>$ chosen by the synthesis process is reorganized in the grasp homogeneous matrix as

$$T_G = \begin{bmatrix} \boldsymbol{s}_G \times \boldsymbol{a}_G & \boldsymbol{s}_G & \boldsymbol{a}_G & \boldsymbol{p}_G \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad , \tag{6.2}$$

the grasp matrix that should be considered at execution time in order to restrict the odometry error and to account for possible movements of the object, can be computed as

$$T_E = T_n^{n-1} \ldots T_2^1 T_1^0 T_G = \left(\prod_{i=0}^{n} T_{n-i}^{n-1-i} \right) T_G \quad , \tag{6.3}$$

with $t_0 = t_G$ and $t_n = t_E$ (See Figure 6.3).

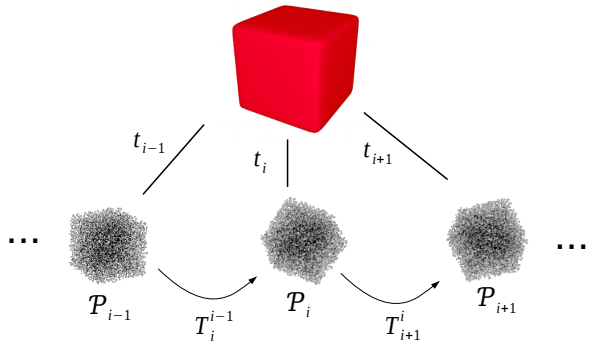
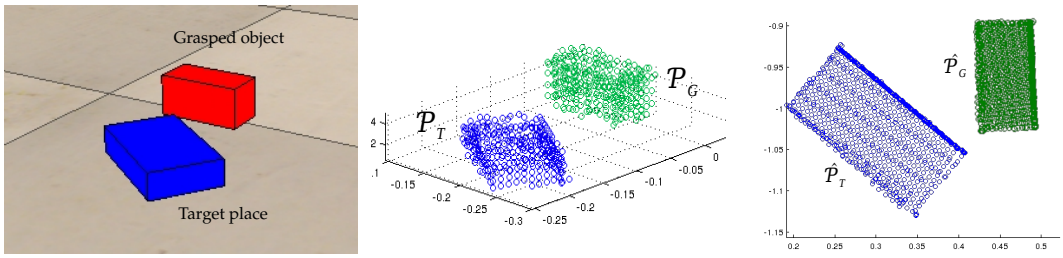


Figure 6.3 Relation between point clouds and homogeneous transformations.

6.4 Placing algorithm

A typical issue, concerning pick and place operations, is to decide how the grasped object should be arranged on the target place. In fact, configurations that allow a stable placement of the object should be preferred, while configurations that may cause the object to topple must be rejected. A placing algorithm has been developed to choose stable placing configurations, based on the hypothesis that the target place has a planar surface on its top.

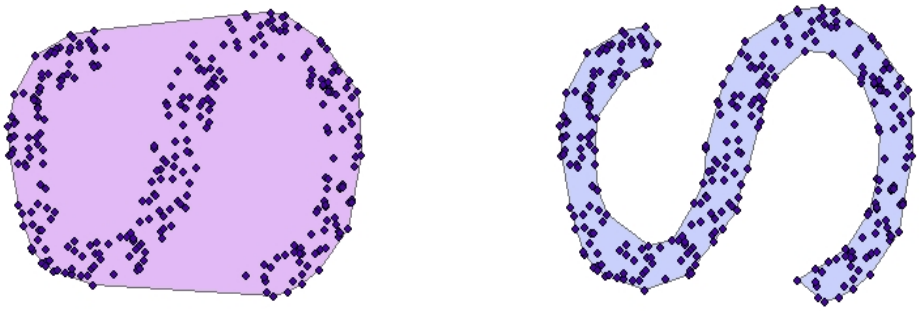
Let $\mathcal{P}_G$ be the point cloud of the grasped object acquired at the last time instant before it has been picked up, and $\mathcal{P}_T$ the point cloud of the target place, i.e. an object on top of which the grasped object should be placed. The bi-dimensional point cloud $\hat{\mathcal{P}}_T$ contains the x and y coordinates of the points belonging to the top surface of the target object, extracted thresholding on all the points heigh. Instead the bi-dimensional point cloud $\hat{\mathcal{P}}_G$ is derived from $\mathcal{P}_G$ considering only the x and y coordinates of all its points (see Figure 6.4).



Grasped object and target place 3D Point clouds 2D Point clouds

Figure 6.4 3D and 2D point clouds considered for the placing problem

The placing problem can be formulated as finding the planar rigid transformation that, applied to the points of $\hat{\mathcal{P}}_G$, places the maximum number of them inside the hull of $\hat{\mathcal{P}}_T$, a polygon that encircles all the points of $\hat{\mathcal{P}}_T$. To define the polygon that best approximates $\hat{\mathcal{P}}_T$, the well known concept of convex hull can be used, as well as more complex approaches, as non convex hulls [48], that better approximate the boundary of the object shape even in case of concave outlines (See Figure 6.5).



Convex hull Non convex hull

Figure 6.5 **Examples of convex and concave hulls.**

In order to formalize the placing problem, it is useful to introduce the function

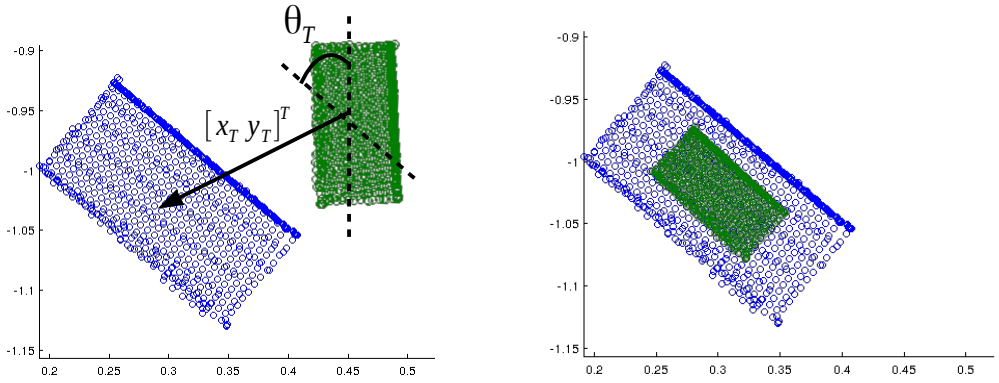
$$insidePolygon(\boldsymbol{p},P)=\begin{cases}1 & if\ \boldsymbol{p}\ inside\ P\\ 0 & otherwise\end{cases} \tag{6.4}$$

which determines if a point $\boldsymbol{p}$ is inside the polygon P , and can be implemented through the *Ray Casting Algorithm* [49].
To superimpose the two point clouds $\hat{\mathcal{P}}_T$ and $\hat{\mathcal{P}}_G$, a roto-translation has been adopted to rigidly transform the generic point $\boldsymbol{p}$ in $\boldsymbol{p}'$ as

$$\boldsymbol{p}'=\begin{bmatrix}\cos(\theta_T) & -\sin(\theta_T)\\ \sin(\theta_T) & \cos(\theta_T)\end{bmatrix}(\boldsymbol{p}-\hat{\boldsymbol{p}}_G)+\begin{bmatrix}x_T\\ y_T\end{bmatrix}+\hat{\boldsymbol{p}}_T \quad , \tag{6.5}$$

where parameter θ_T expresses the transformation rotational angle, parameters x_T and y_T are the transformation translational components,

$\hat{\boldsymbol{p}}_{\boldsymbol{G}}$ is the centroid of point cloud $\hat{\mathcal{P}}_G$, $\hat{\boldsymbol{p}}_{\boldsymbol{T}}$ is the centroid of point cloud $\hat{\mathcal{P}}_T$.



Initial configuration Optimal placing configuration

Figure 6.6 **Example of optimal placing problem.**

At this point, the placing optimization problem can be formalized as

$$\underset{\theta_T, x_T, y_T}{argmax} \sum_{\boldsymbol{p}' \in \hat{\mathcal{P}}'_G} insidePolygon\left(\boldsymbol{p}', Hull\left(\hat{\mathcal{P}}_T\right)\right) \tag{6.6}$$

where $\hat{\mathcal{P}}'_G$ is the set of all the points of $\hat{\mathcal{P}}_G$ transformed by the roto-translation having parameters θ_T , x_T and y_T , while the term $Hull\left(\hat{\mathcal{P}}_T\right)$ represents the polygon used to approximate the contour of $\hat{\mathcal{P}}_T$. The optimization problem has been solved in this work with a sample based algorithm, which first searches for the rotation that best superimposes the two point clouds, and then searches for the translational components of the transformation. The placing algorithm pseudo-code will be presented here in detail.

Placing algorithm

Input:

- Grasping pose $G = <\boldsymbol{p}_G, \boldsymbol{s}_G, \boldsymbol{a}_G, l_1>$ used to pick up the goal object.
- 2D point cloud $\hat{\mathcal{P}}_G$ of the grasped object.
- 2D point cloud $\hat{\mathcal{P}}_T$ of the target place top surface.
- Centroid $\hat{\boldsymbol{p}}_G$ of the grasped object.
- Centroid $\hat{\boldsymbol{p}}_T$ of the target place top surface.
- Number of translational samples K_{max} .
- Number of rotational samples θ_{max} .
- Height of the target surface z_T .

Output:

- Optimal placing pose G_T .

1. **For** $\theta = 0 : \frac{2\pi}{\theta_{max}} : 2\pi$ *(Rotation)*
2. $f_{max} = 0$
3. $\hat{\mathcal{P}}'_G = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} (\hat{\mathcal{P}}_G - \hat{\boldsymbol{p}}_G) + \hat{\boldsymbol{p}}_T$
4. **If** $\sum_{\boldsymbol{p}' \in \hat{\mathcal{P}}'_G} insidePolygon(\boldsymbol{p}', Hull(\hat{\mathcal{P}}_T)) > f_{max}$
5. $\theta_T = \theta$
6. $f_{max} = f$
7. **End**
8. **End**
9. **For** $k = 0 : 1 : K_{max}$ *(Translation)*
10. $\boldsymbol{d}_{rand} = random\ direction$
11. $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_T \\ y_T \end{bmatrix} + \epsilon\, \boldsymbol{d}_{rand}$
12. **Do**
13. $ok = true$
14. $\hat{\mathcal{P}}'_G = \begin{bmatrix} \cos(\theta_T) & -\sin(\theta_T) \\ \sin(\theta_T) & \cos(\theta_T) \end{bmatrix} (\hat{\mathcal{P}}_G - \hat{\boldsymbol{p}}_G) + \begin{bmatrix} x \\ y \end{bmatrix} + \hat{\boldsymbol{p}}_T$

15. **If** $f > f_{max}$
16. $x_T = x$
    $y_T = y$
17. $f_{max} = f$
18. **Else**
19. $ok = false$
20. **End**
21. **While** ok
22. **End**
23. $$R_z(\theta_T) = \begin{bmatrix} \cos(\theta_T) & -\sin(\theta_T) & 0 \\ \sin(\theta_T) & \cos(\theta_T) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
24. $$G_T = < R_z(\theta_T)(\boldsymbol{p_G} - \hat{\boldsymbol{p}}_G) + \hat{\boldsymbol{p}}_T + [x_T, y_T, 0]^T, R_z(\theta_T)\boldsymbol{s_G}, R_z(\theta_T)\boldsymbol{a_G}, l_1 >$$

6.5 Software architecture

The software architecture of the developed framework has been designed so as to make the system modular, reusable and maintainable (see Figure 6.7).

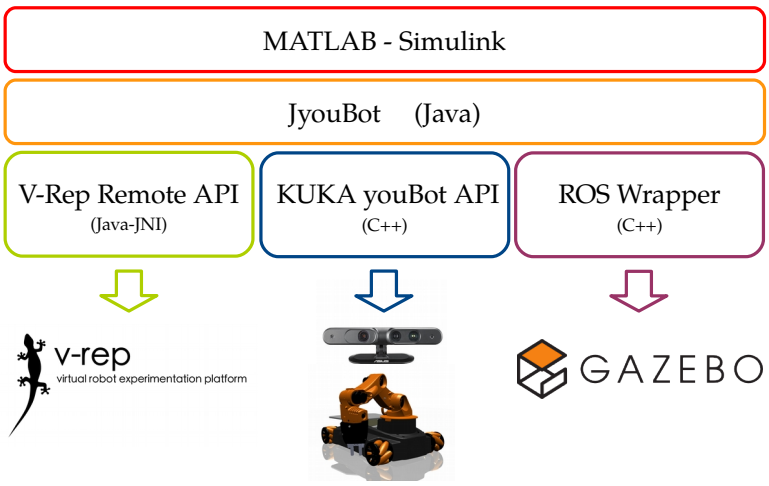


Figure 6.7 **Software architecture.**

The main software module, containing the application logic and all the control algorithms presented throughout this work, has been implemented within the well known *MATLAB - Simulink* graphical tool [50]. Then the middleware layer *JyouBot* has been realized to expose a unique interface, which defines the low level functions and services necessary to control the robot actuation and to receive informations from the sensor devices, as the motors encoders, the proximity sensors and the depth camera (see Appendix B). The JyouBot interface allows to maintain the same implementation of the application logic to control the whole framework in different situations, as for example in different simulation environments or in the real case. Indeed, by implementing a particular feature of the JyouBot interface, one can customize the developed control system of the youBot robot to work in the desired simulated or real environment with the available sensor devices. Three different implementations of the JyouBot services have been realized. The first one allows to perform simulations in the V-Rep environment [51] using the remote API provided by the simulation framework itself. The already available model of the KUKA youBot has been used for the simulation experiments, as well as six virtual proximity sensors and a virtual depth camera placed on the model of the robot platform. The second implementation, which uses the standard KUKA youBot API [52], has been developed to execute experimental testing directly on the real KUKA youBot robot. Moreover, the common OpenNI library [53] has been adopted to acquire the visual images from the ASUS Xtion device [54], which has been used both to reconstruct the model of the objects to be manipulated and to acquire the structure of the environment necessary for the robot navigation, as if it was a proximity sensor. Finally the ROS wrapper [55] for the KUKA youBot has been adopted in a third implementation to perform simulations in the Gazebo [56] environment.

Chapter 7

Experimental results

In this section some experimental results, concerning the thematics proposed in the previous chapters, will be presented.

7.1 Internal motions

In order to evaluate the capability of the developed framework to correctly control the motion of the redundant KUKA youBot, an experiment has been conducted to observe the robot internal motions. Internal motions are a typical feature of redundant manipulators, which can change their state and move their mechanical structure without changing the end-effector position and orientation. This behavior has been tested on the KUKA youBot by observing the positional error of the end-effector with respect to a fixed desired configuration, while the redundancy parameters ρ_1 , ρ_2 and ρ_4 vary with time. During the experiment, the robot end-effector desired configuration was set to $\boldsymbol{X}=[0\ 0\ 0.1\ \pi\ 0\ 0]^T$, while the values of the redundancy parameters were first varied one at a time, and then, at the end, all together. The result of the experiments are shown in Figure 7.1, where the evolution during time of the end-effector positional error and the varying values of the redundancy parameter are illustrated. This experiment was executed using the built-in joint position control of the KUKA youBot, while the results shown in Figure 7.2 refer to the same experiment executed instead with the built-in joint velocity control. As already demonstrated in [57], the achieved results show that the position control can lead to a very accurate positioning of the end-effector, but causes a slight undesired vibration of the manipulator structure. The joint velocity control, instead, allows to follow a desired end-effector trajectory in a smoother way, but it presents a major positioning error with respect to the position control. The same experiments regarding internal motions, with both the position and velocity control, were repeated imposing to the end-effector to follow a linear Cartesian trajectory $\boldsymbol{X}=[k_1 t\ 0\ k_2 t\ \pi\ 0\ 0]^T$ with $k_1=0.1$ and $k_2=0.01$ (See Figure 7.3 and 7.4).

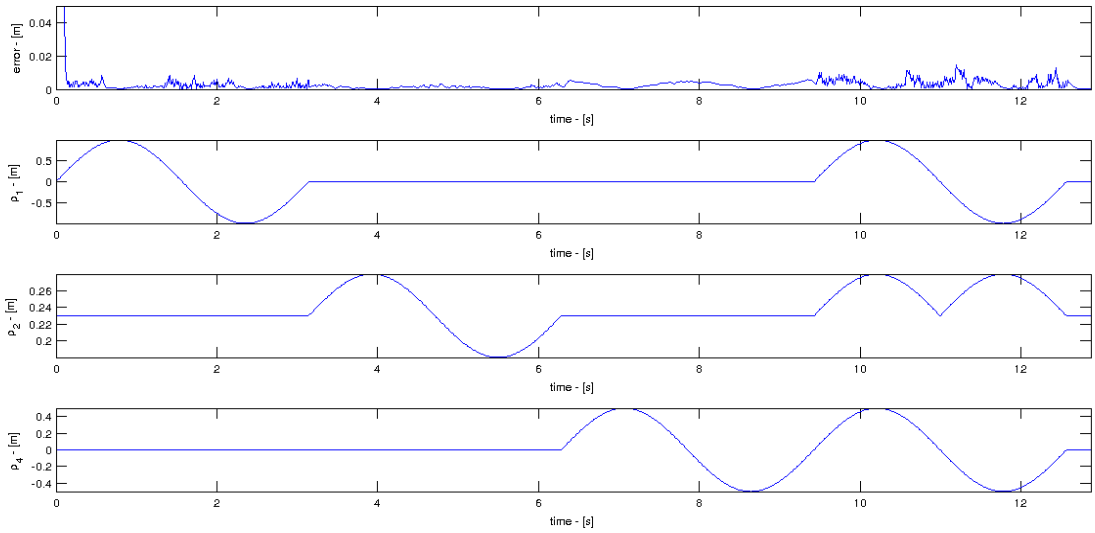


Figure 7.1 End-effector positional error during internal motion (position control).

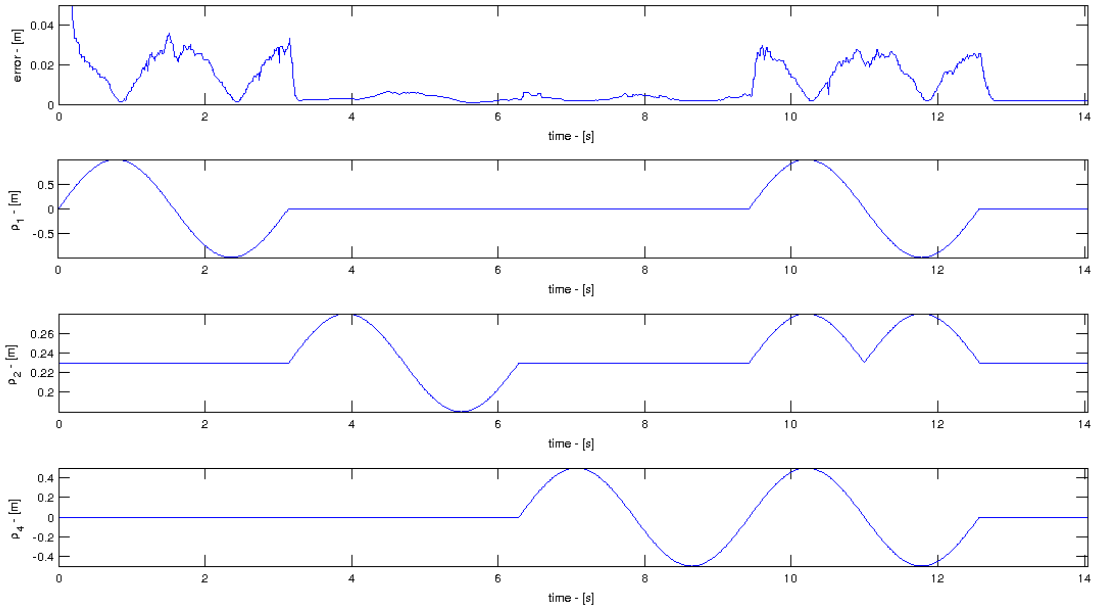


Figure 7.2 End-effector positional error during internal motion (velocity control).

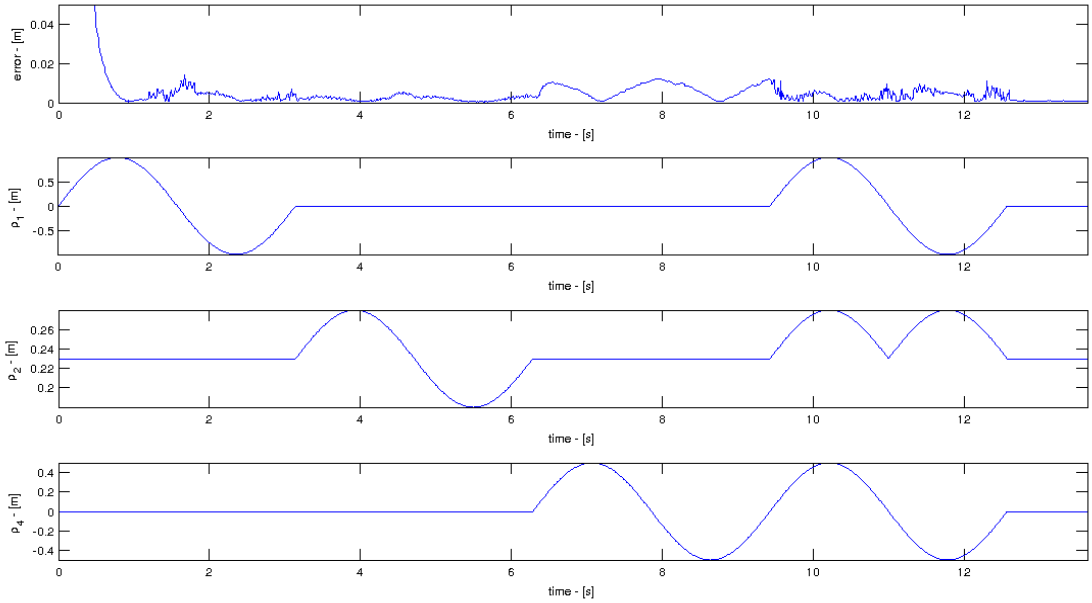


Figure 7.3 **End-effector positional error during internal motion while following a linear Cartesian trajectory (position control).**

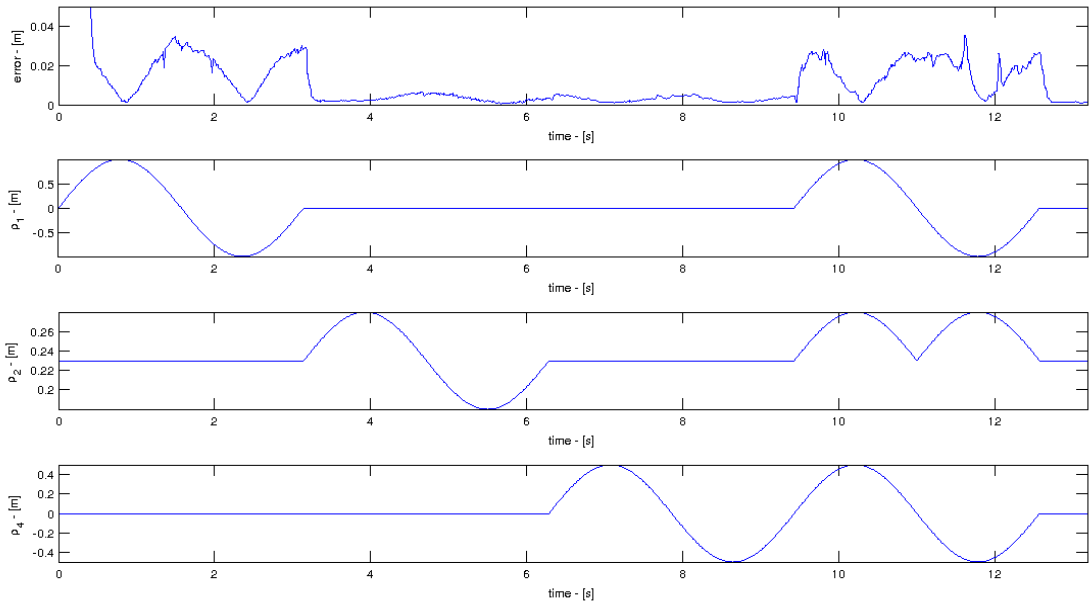


Figure 7.4 **End-effector positional error during internal motion while following a linear Cartesian trajectory (velocity control).**

These results show the ability of the developed controller to manage the eight degree of freedom of the KUKA youBot robot, when both the Cartesian input variables and the redundancy parameter vary with time. In fact, the end-effector positional error, expressed in the figures as an euclidean norm, is always less the 1 cm, when the position control is used, and is less then 2 cm, when the velocity control is used instead.

7.2 Manipulability optimization

The arm extension redundancy, described by parameter ρ_2 , has been used to maximize the index $U(\boldsymbol{q})$ with the double objective of both keeping the joints of the manipulator away from their physical limits and maximizing the manipulability index, as presented in Chapter 3. The optimization, executed on-line during the manipulation tasks, actually succeeded in the objective of choosing configurations of the mobile manipulator suitable for objects manipulation. Figure 7.5 and 7.6 show the values achieved by the objective function $U(\boldsymbol{q})$ for each possible value of redundancy parameter ρ_2 , during a pick and place operation. The black line indicates the actual values of ρ_2 selected during the task execution. It is possible to note that, after an initially transient the optimization strategy succeeds in choosing values of ρ_2 that maximize the objective measure $U(\boldsymbol{q})$.

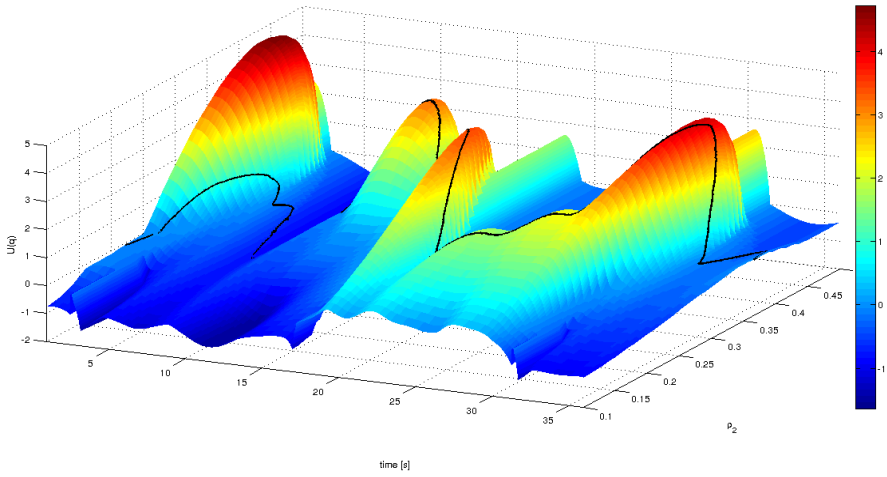


Figure 7.5 **Behaviour of objective function $U(\boldsymbol{q})$ and chosen values of redundancy parameter ρ_2 during a pick and place task.**

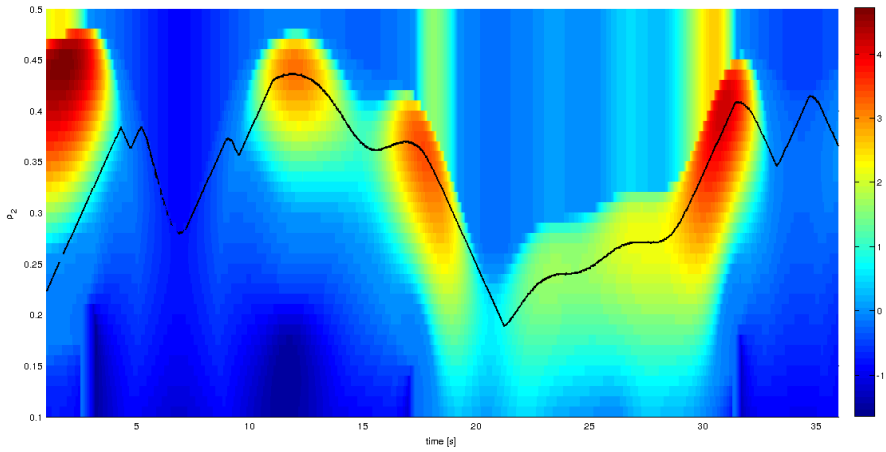
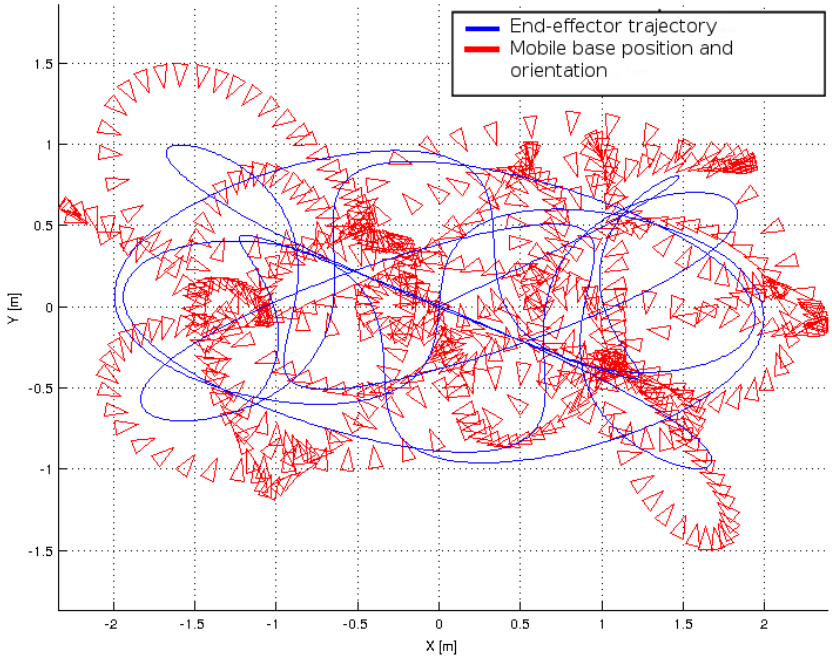


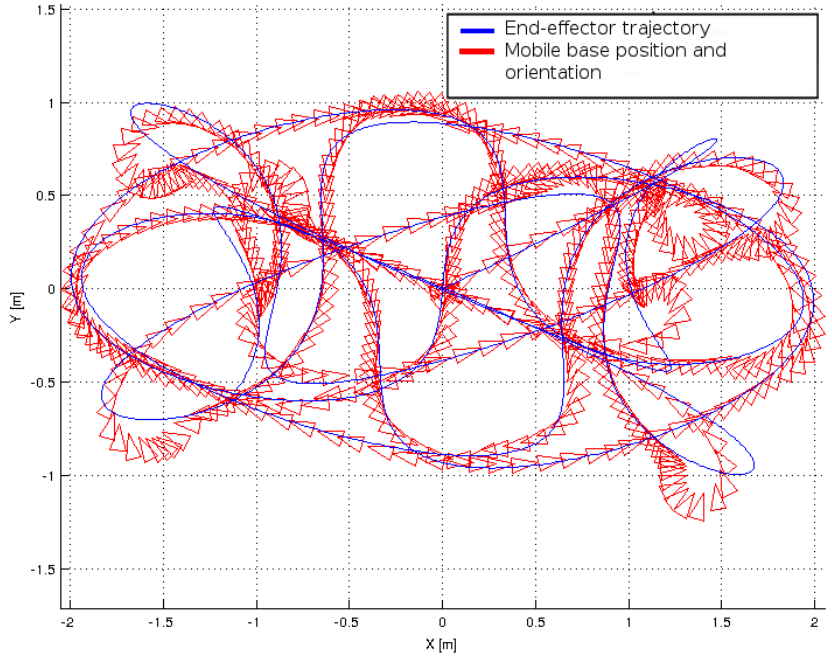
Figure 7.6 **Behaviour of objective function $U(\boldsymbol{q})$ and chosen values of redundancy parameter ρ_2 during a pick and place task. (Heat-map)**

7.3 Vertical end-effector configuration redundancy

As described in Chapter 3, the redundancy parameter ρ_4 has been introduced to describe a task redundancy that occurs when the end-effector of the KUKA youBot is in a vertical configuration. This redundancy has been exploited to minimize the movements of the mobile base. To observe the effectiveness of the developed redundancy resolution strategy, an experiment was performed, where the youBot follows a Cartesian trajectory for one minute always keeping a vertical posture of its end-effector. The experiment was executed twice, in both cases with the same Cartesian trajectory. First the behavior of the robot was tested when no redundancy resolution strategy is actuated on parameter ρ_4 , then the experiment was repeated considering the developed redundancy resolution strategy. As shown in Figure 7.7, when the redundancy resolution strategy has been considered, the mobile base movements were reduced by 22,5 % compared to the other case (from 62.1 m to 48.1 m) .



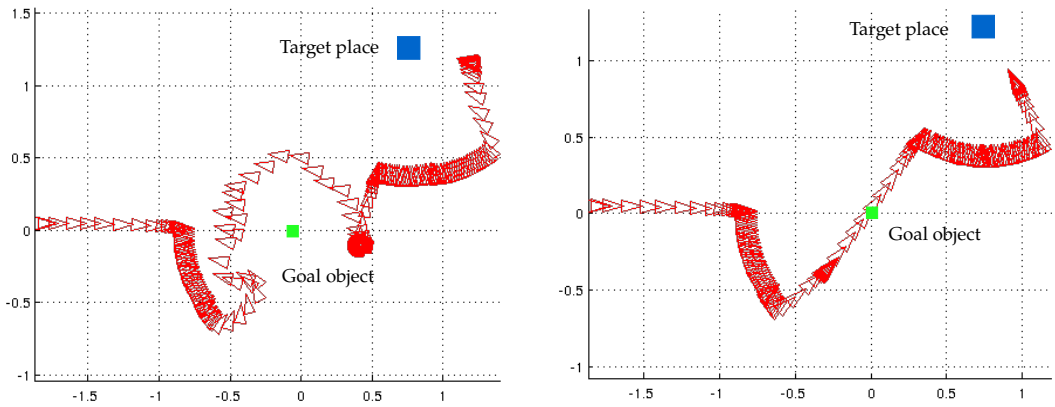
No vertical end-effector redundancy resolution (Total mobile base movements: 62.1 m).



Vertical end-effector redundancy resolution (Total mobile base movements: 48.1 m).

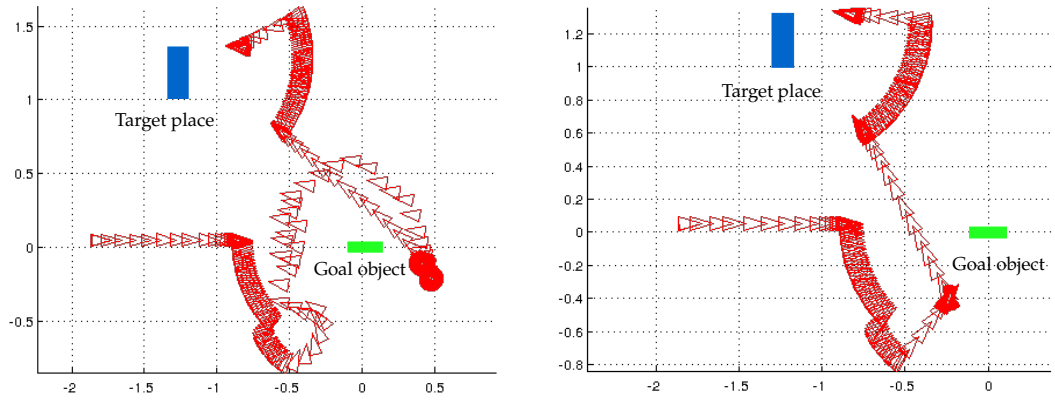
Figure 7.7 Mobile base position and orientation while following a Cartesian end-effector trajectory with a fixed vertical configuration of the end-effector.

Then, the movements of the mobile base of the robot were observed during some pick and place operations. The same pick and place task was repeated both with and without the developed redundancy resolution strategy for the end-effector configurations. As illustrated in Figures 7.8 and 7.9, even in this case the mobile base movements are reduced when the task redundancy of the robot is considered. Indeed, if no redundancy resolution strategy is considered, the last joint of the youBot manipulator does not execute any rotational compensation and the mobile base must perform an additional movement to align the arm with the selected grasp configuration. In Figure 7.8 and 7.9 the object that has to be grasped is colored in green, while the target place in blue.



No vertical end-effector redundancy resolution Vertical end-effector redundancy resolution

Figure 7.8 Mobile base position and orientation during a pick and place operation.



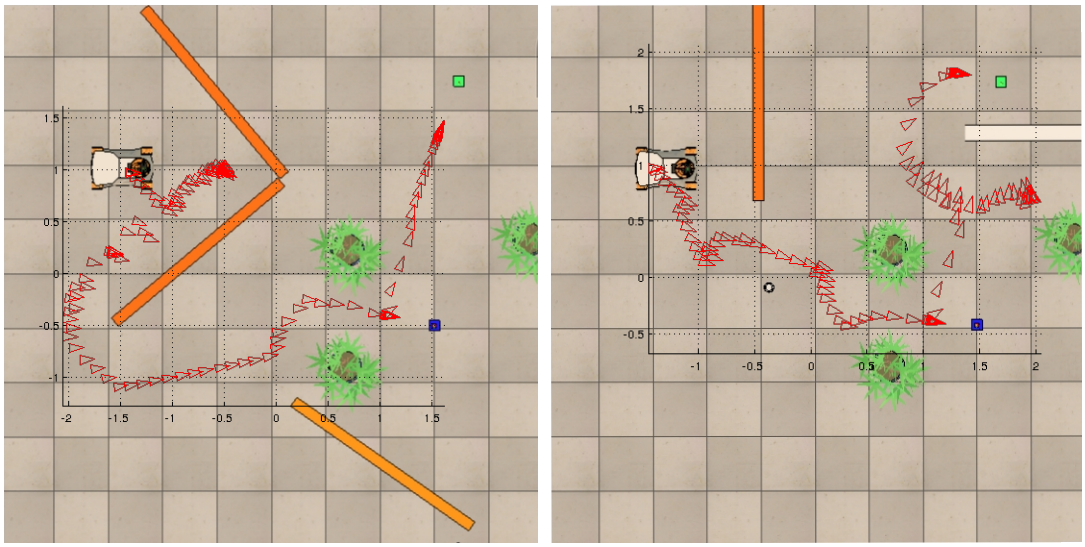
No vertical end-effector redundancy resolution Vertical end-effector redundancy resolution

Figure 7.9 Mobile base position and orientation during another pick and place operation.

7.4 Navigation

The navigation technique based on rotational potential fields, introduced in Chapter 4, has been tested in the VRep simulation environment using six virtual proximity sensors mounted around the mobile base of the KUKA youBot robot to perceive obstacles. As illustrated in Figure 7.10, the robot was able to safely navigate in the environment while avoiding obstacles. As expected, the introduction of rotational potential fields ensures the navigation to be not affected by the problem of local minima in the potential function, typical of standard potential field techniques. Indeed the robot was able to navigate in a cluttered environment with many obstacles, whose position and orientation are not known in advance. Also the case of obstacles with a concave outline, which result more difficult to bypass by a reactive navigation technique, was examined with positive results as shown in Figure 7.10 and in the accompanying video [58].

Thanks to the modular architecture of the system, it has been possible to test the developed navigation strategy also on the real KUKA youBot, in this case using the ASUS Xtion device to perceive the environment structure, instead of the proximity sensors used during the simulation experiments. Also in the real case the youBot was able to correctly navigate in the laboratory, as shown in Figure 7.11 and in the accompanying video [59].



Trap shape obstacle Narrow passages in a cluttered environment

Figure 7.10 Mobile base movements during navigation in a cluttered environment in VRep simulation environment.

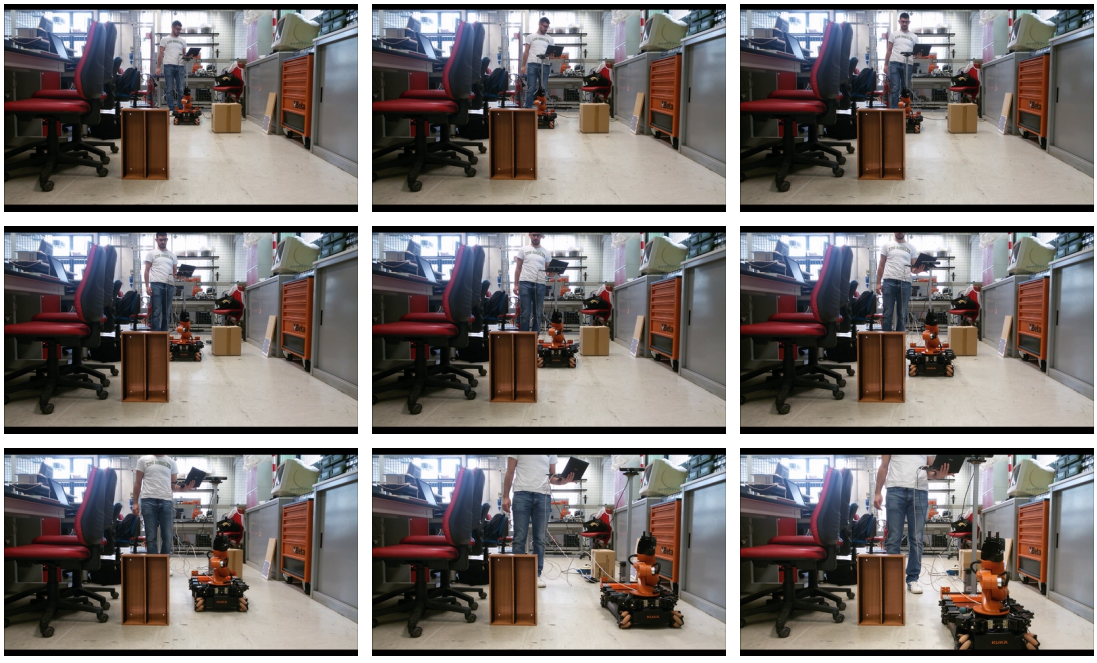


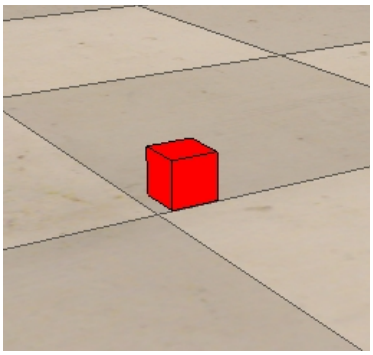
Figure 7.11 Navigation experiment with the real KUKA youBot.

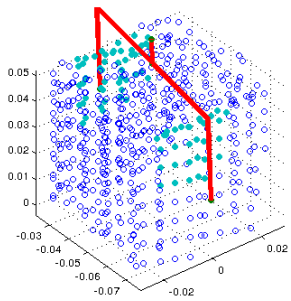
7.5 Grasp synthesis

The grasp synthesis process, described in Chapter 5, has been initially tested in the VRep simulation environment, where a virtual depth camera sensor has been inserted in the model of the robot, fixed on its mobile platform. The outcome of the grasp synthesis process has been analyzed during the execution of various pick and place operations, where shape, position, orientation and dimensions of the objects to be manipulated are a-priori unknown to the robot. At the beginning, some objects with different shapes were placed on the floor and the grasp synthesis process always correctly selected a vertical grasping configuration, because lateral end-effector ones are not achievable in this situation with the small KUKA youBot arm, as shown in the following test cases. Such test cases report the real object in the simulation environment with the corresponding point cloud model reconstructed by the synthesis process and the selected grasp pose. The execution time reported in the different test cases, which have been executed on a common low cost laptop with an Intel Core I3 processor, includes all the four stages of the grasp synthesis process described in Chapter 5: Point Cloud Acquisition, Data Pre-processing, Grasp Generation, Grasp Selection.

Test case 1 – Small cube

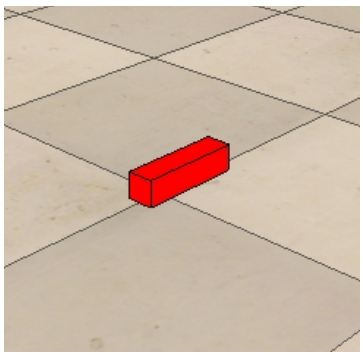
Computation time: 0.295 s, **Number of admissible generated grasp poses:** 258.

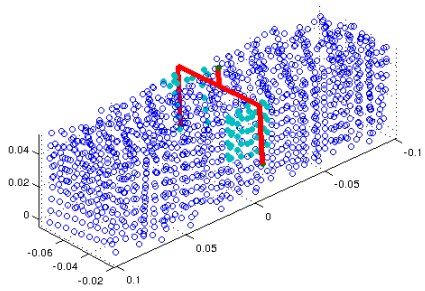




Test case 2 – Bar

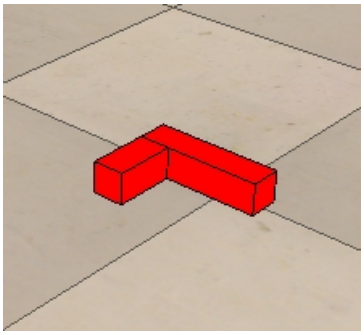
Computation time: 1.04 s , **Number of admissible generated grasp poses:** 312.

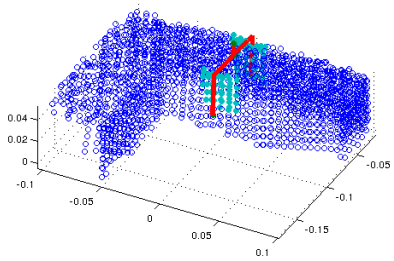




Test case 3 – Concave object

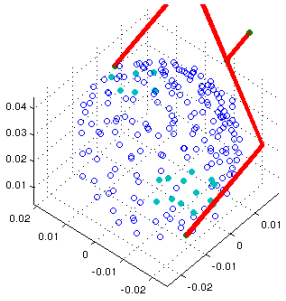
Computation time: 1.39s, **Number of admissible generated grasp poses:** 417.





Test case 4 – Small sphere

Computation time: 0.17s, Number of admissible generated grasp poses: 646.

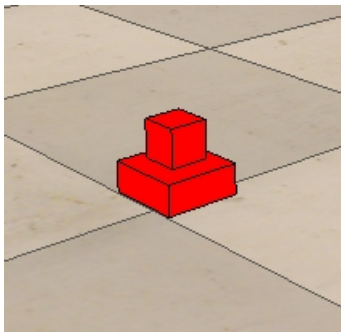


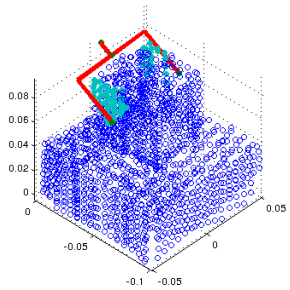
Afterwards, bigger objects as well as smaller ones placed on an elevated surface were considered, as they could be grasped even with lateral configuration of the end-effector. Also in this case the synthesis process

correctly generated admissible grasp poses, as proved by the following test cases.

Test case 5 – Concave big object

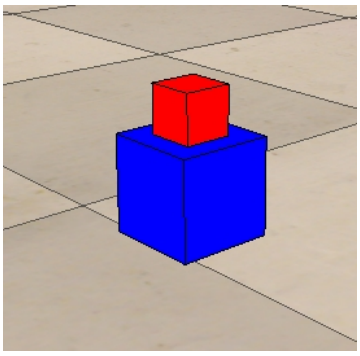
Computation time: 0.87s, **Number of admissible generated grasp poses:** 30.

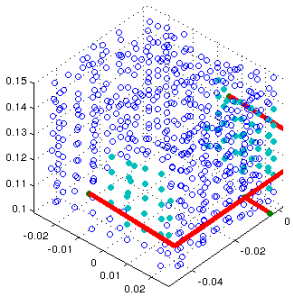




Test case 6 – Cube on cube

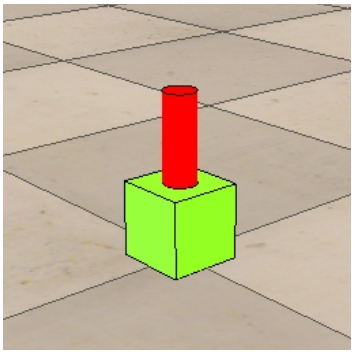
Computation time: 0.485 s, **Number of admissible generated grasp poses:** 688.

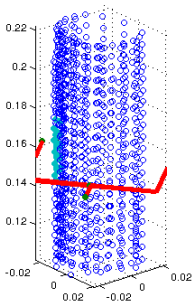




Test case 7 – Cylinder on cube

Computation time: 0.84 s, **Number of admissible generated grasp poses:** 873.

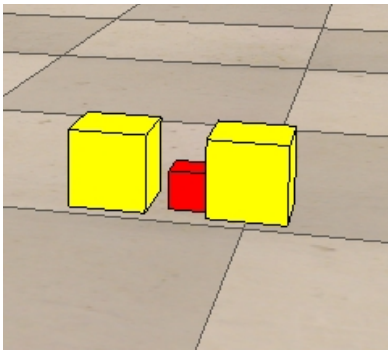


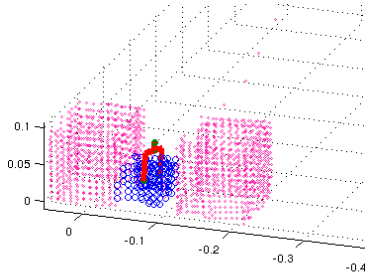


Finally, the case of particularly cluttered environment, where other obstacles are positioned near the target object has been considered. The synthesis process was able to identify grasping configurations that allow to execute the grasping task avoiding collision with the close obstacles.

Test case 8 – Cube between obstacles

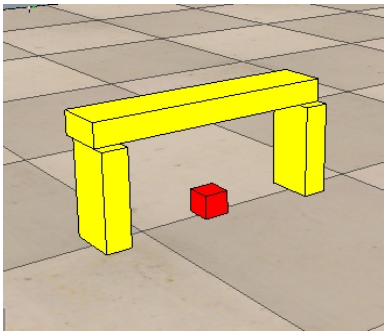
Computation time: 0.08 s, **Number of admissible generated grasp poses:** 19.

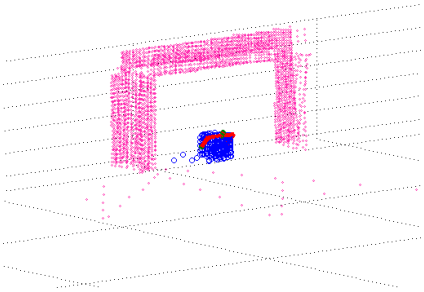




Test case 9 – Cube under a bench

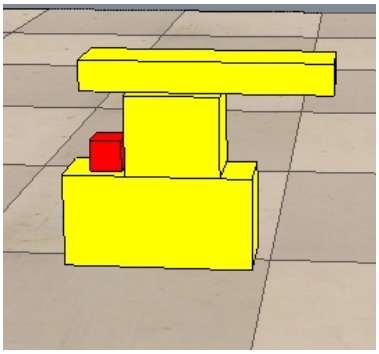
Computation time: 0.15 s, **Number of admissible generated grasp poses:** 7.

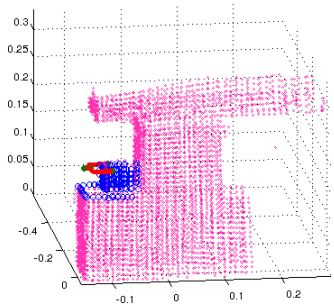




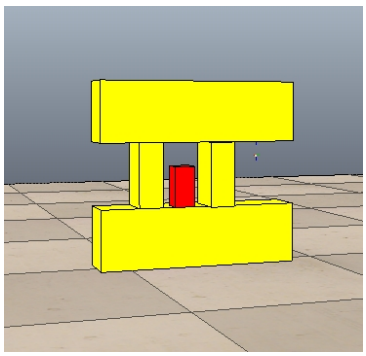
Test case 10 – Cube on a shelf

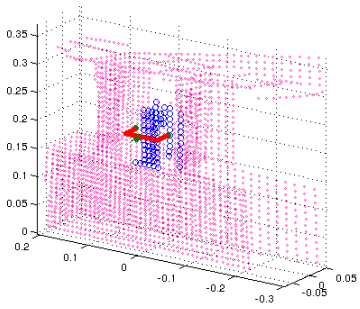
Computation time: 0.088 s, **Number of admissible generated grasp poses:** 5.





Test case 11 – Prism on a shelf
Computation time: 0.07s , **Number of admissible generated grasp poses:** 21.





In general, the pick and place framework proved to be particularly robust during the simulations experiments, as every requested task was correctly accomplished by the robot. As shown by the previous test cases and the in the accompanying video [60], the grasp synthesis process is very effective and produces stable and intuitive grasp postures, requiring less then a second of computation time on average. The testing conducted on the grasp synthesis algorithm in the simulation environment are generally valid even in the real case, because they are based only on the point cloud model of the objects, and the current available depth cameras are even more accurate than the virtual sensors used in simulation.

7.6 Pick and place operations with the real KUKA youBot

The overall behavior of the robotic system was tested on a real KUKA youBot mobile manipulator. The ASUS Xtion Live Pro device was mounted on the robot platform and the common OpenNI library was adopted to acquire the RGB images and the depth scansions. The data acquisition phase of the grasp synthesis process was particularly critical, as the perceived color of the objects is sensitive to the light of the laboratory that changes considerably with the outside weather and with the time of the day. For this reason the OpenNI library was used to develop a simple tool that allows to manually calibrate the color filter in few seconds. The application allows to select the ranges of hue, saturation and lightness and shows the filtering results on the acquired images, making it possible to quickly select the color parameters of the different objects to be manipulated (see Figure 7.12).

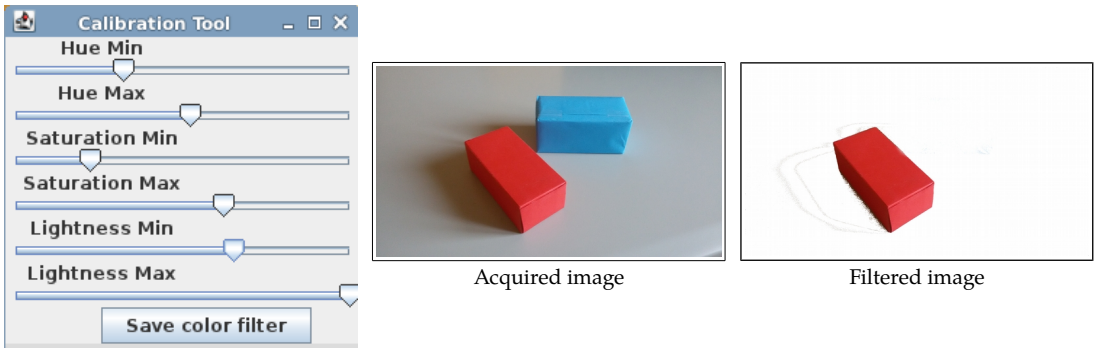
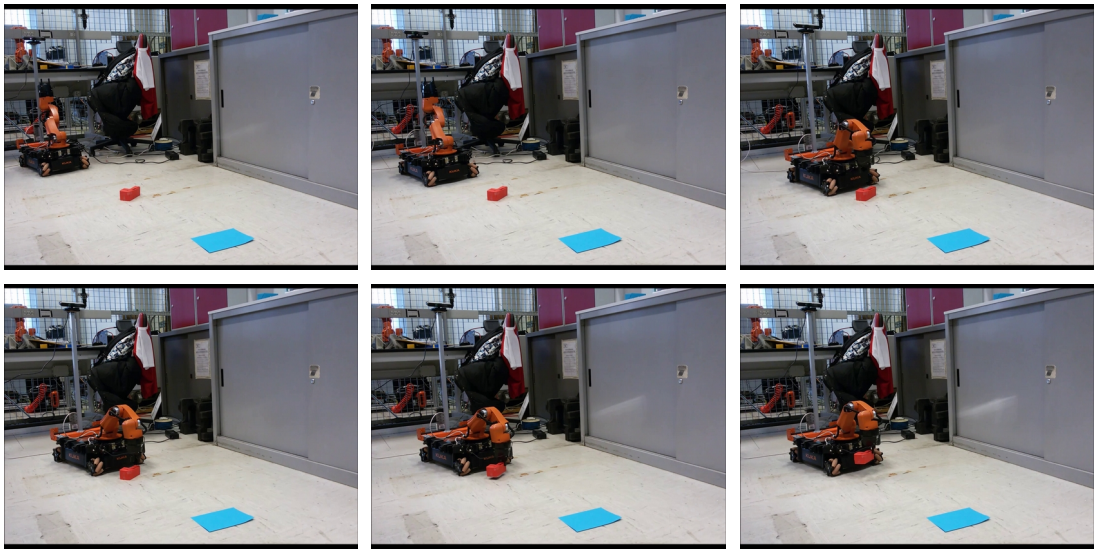


Figure 7.12 **Colour filter calibration tool.**

Furthermore, a method to automatically calibrate the position of the camera with respect to the robot frame has been implemented based on a linear regression applied to the floor observations (see Appendix C). With these tools implemented, the pick and place operations were tested with satisfactory results, as shown in Figure 7.13 and in the accompanying video [59]. The robot was able to autonomously localize the objects, reconstruct their point cloud model, correctly perform the grasp synthesis, plan and execute the pick and place task. In some cases, the grasping execution failed, mainly because of the standard youBot gripper that was used, that has a small stroke and limited grasping forces.



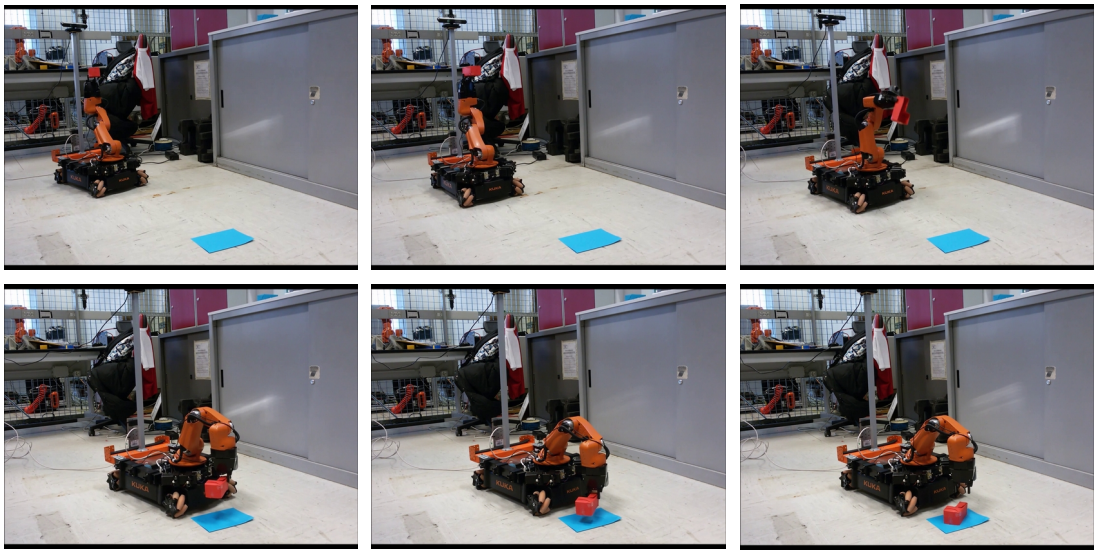


Figure 7.13 **Pick and place experiment with the real KUKA youBot.**

The pick and place framework was tested with positive results on a small prism, on a concave object with few admissible grasp postures and on rubber ball (see Figure 7.14).

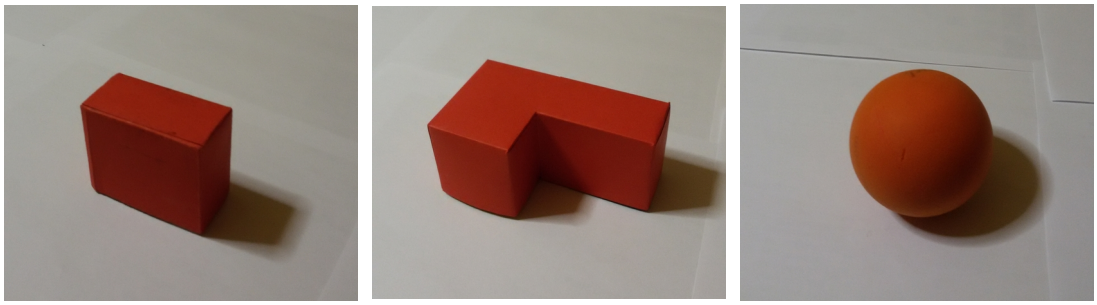


Figure 7.14 **Object used for manipulation experiments on the real robot.**

Conclusions and future works

The aim of this thesis was to develop control techniques and algorithms, which would allow to perform robotic manipulation tasks in completely unknown environment, exploiting the functional flexibility of mobile manipulators.

An autonomous pick and place framework has been successfully designed and implemented on a KUKA youBot research platform. Thanks to the realized application, the robot is able to autonomously explore the environment, reconstruct the model of the objects that should be grasped, plan and correctly execute pick and place operations.

The benefit of introducing robot specific redundancy parameters has been demonstrated, in this thesis, by showing that each exceeding degree of freedom can be coupled with a specific desired behavior enforced by the chosen redundancy resolution strategy. In this way, the redundancy of the robot is not used to optimize a unique performance measure, but each degree of redundancy accomplishes a specific behavior, based on what the robot control designer thinks it can do best. The developed redundancy resolution strategies allow the youBot robot to keep its joints inside their physical limits, maximize the whole robot manipulability index, improve the coordination between mobile base and arm, reduce the movements of the mobile platform, keep a certain object inside the field of view of the vision subsystem.

Although not formally proven, the developed navigation technique based on rotational potential field results to be local minima free in all the numerous test cases taken in consideration, where the robot is correctly guided toward its goal, while avoiding collisions with obstacles disposed inside the environment.

The very interesting results, obtained by the original grasp synthesis process realized in this work, show that on-the-fly grasping techniques can be robust as common recognition-based ones, but turn out to be definitely more computationally efficient and more suitable to deal with unknown objects and environments.

In conclusion, the achieved results are quite satisfactory as all the indicated objectives have been attained.

During the evolution of this work, many ideas for further researches and possible applications have been taken into consideration, although there has been no time to develop them in a concrete and complete way. Here some possible guidelines for future work will be shortly presented.

Besides pick-and-place tasks, it would be very interesting to study how mobile manipulator could perform autonomous assembly operations. As in the well know case of peg in hole insertion, assembly tasks require more advanced control techniques like *force control*, which takes into consideration also the dynamic behavior of the robot. Torque control has been used in this work to perform a compliant manual guidance of the youBot arm, as described in Appendix D. Future researches could investigate how to obtain a dynamic model of the whole mobile manipulator, considering also the effects of the mobile platform, and use it to develop control strategies for assembly operations.

A simple smart-phone application has been implemented to control the youBot behavior with vocal commands. In order to improve human-robot interaction aspects, further studies could be used to integrate mobile manipulators with other devices as smart-phones, advanced vision devices, Enterprise Resource Planning systems or Domotic environments.

Appendix A

KUKA youBot hardware specification

A.1 Arm technical data

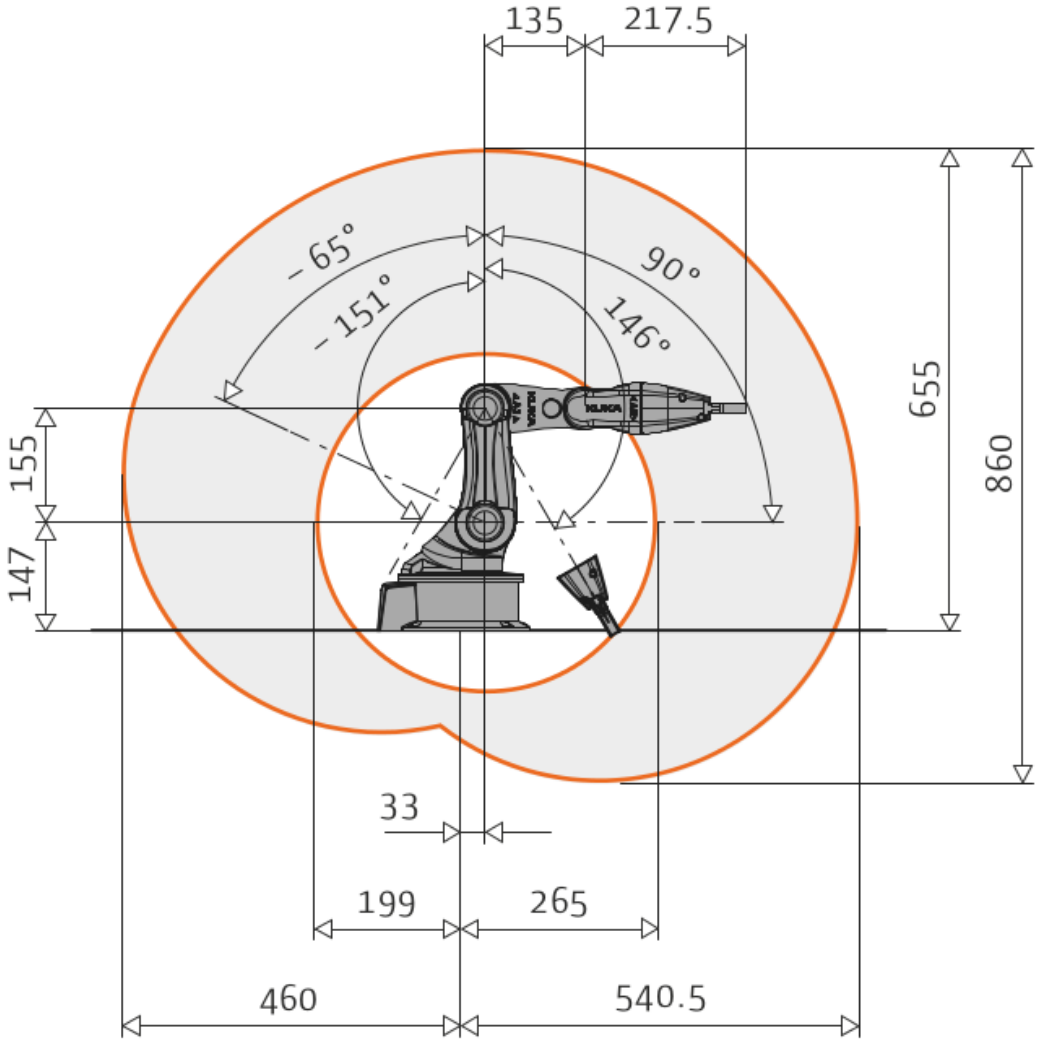


Figure A.1 **KUKA youBot Arm specification**

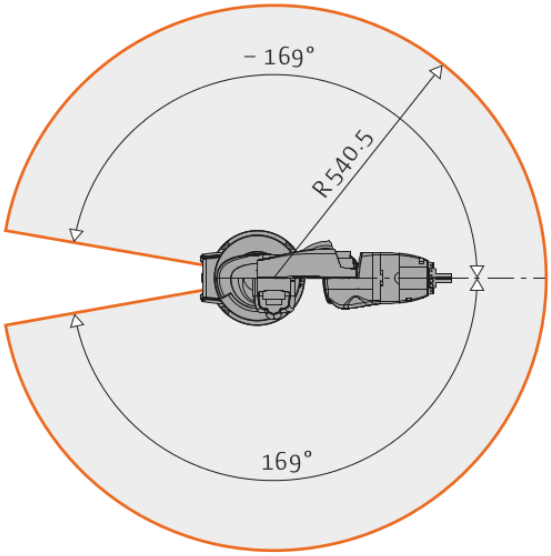


Figure A.2 **KUKA youBot Arm specification**

	Lower limit	Upper limit	Speed
Axis 1	-169°	+169°	90 °/s
Axis 2	-65°	+90°	90 °/s
Axis 3	-151°	+146°	90 °/s
Axis 4	-102°	+102°	90 °/s
Axis 5	-167°	+167°	90 °/s

Table A.1 **KUKA youBot Axis specification**

Gripper stroke	20 mm
Gripper range	70 mm
Motors	2 independent stepper motors

Table A.2 **KUKA youBot Gripper specification**

A.2 Base technical data

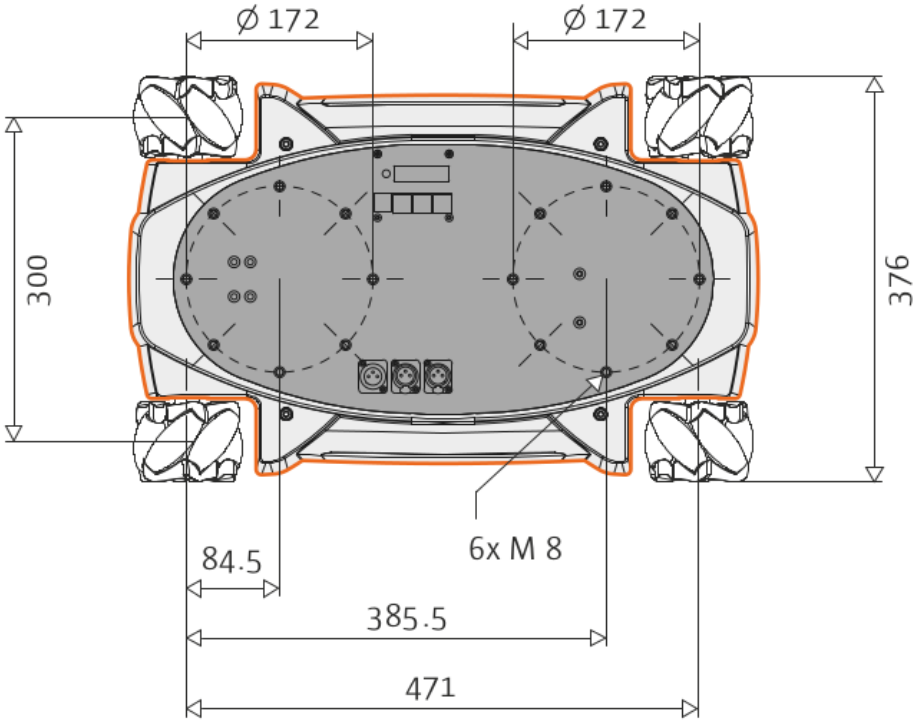


Figure A.3 **KUKA youBot Base specification**

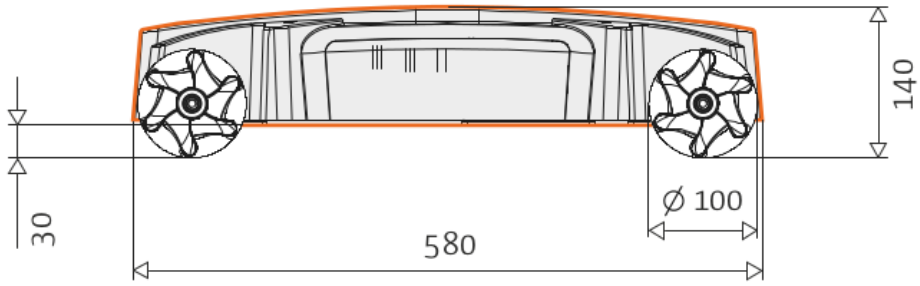


Figure A.4 **KUKA youBot Base specification**

Omnidirectional Kinematics	4 KUKA omniWheel
Speed	0.8 m/s

Table A.3 **KUKA youBot Base specification**

A.3 Denavit-Hartemberg parameters

	θ	d	a	α
Link 1	q_1	0.147	0.033	$\pi/2$
Link 2	q_2	0	0.155	0
Link 3	q_3	0	0.135	0
Link 4	q_4	0	0	$\pi/2$
Link 5	q_5	0.187	0	0

Table A.4 Denavit-Hartenberg parameters for KUKA youBot manipulator

x_d	0.166 m
y_d	0 m
z_d	0.98 m

Table A.5 KUKA youBot manipulator base frame displacement from platform frame

Appendix B

JyouBot interface

The *JyouBot* middleware layer, inside the context of the autonomous pick and place framework presented in Chapter 6, has been developed in this thesis to provide a unique Java interface, which defines the low level functions and services necessary to control the robot actuation and to receive informations from the sensory devices. The JyouBot interface allows to maintain the same implementation of the framework application logic, while customized implementations to support the adopted devices, in real or simulated environment, are still possible. In this way, by implementing a particular feature of the JyouBot interface, one can personalize the developed control system of the youBot robot to work in the desired environment with the available sensory devices. More in detail, the class *JyouBot* exposes the methods that control the velocity of the youBot mobile base and the positions, velocities and torques of the youBot arm joints. The same class can be used to retrieve the joints positions from the motors encoders, and to access to the odometry information that describe the position and orientation of the mobile platform. The class *GoalDetector* defines the methods necessary to acquire the point cloud of the objects that should be manipulated, based on their color. The class *ObstacleDetector* defines the methods devoted to the acquisition of the obstacles positions, necessary for the navigation of the robot. A class diagram of the described interface is reported in Figure B.1.

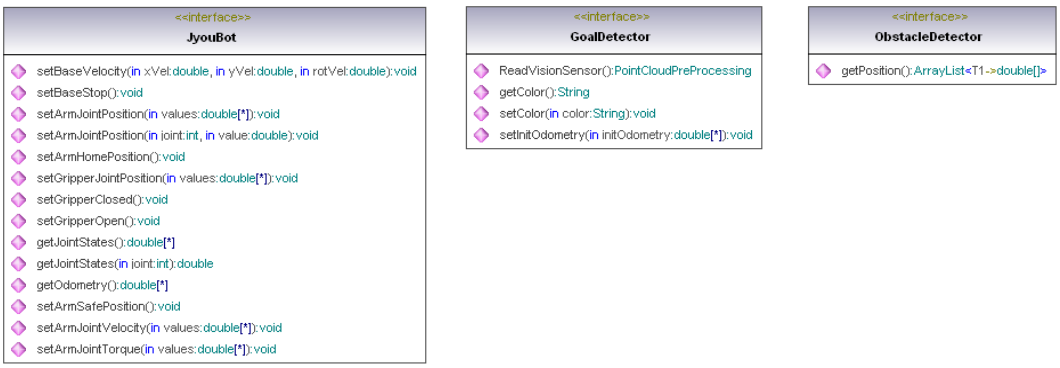


Figure B.1 **Class diagram of JyouBot interface.**

A detailed list of the functionalities defined by the JyouBot interface can be found in the following tables.

JyouBot interface method summary

Modifier and Type	Method and Description
`double[]`	**getJointStates**`()` Get the actual values of arm and gripper joints.
`double`	**getJointStates**`(int joint)` Get the actual value of a specific joint.
`double[]`	**getOdometry**`()` Get the odometry data of the mobile base.
`void`	**setArmHomePosition**`()` Move the manipulator near the home position.
`void`	**setArmJointPosition**`(double[] values)` Set the joint position set-point of the manipulator joints.
`void`	**setArmJointPosition**`(int joint, double value)` Set the joint position set-point of a specific joint of the manipulator.
`void`	**setArmJointTorque**`(double[] values)` Set the joint torque set-point for the manipulator joints.
`void`	**setArmJointVelocity**`(double[] values)` Set the joint velocity set-point for the manipulator joints.
`void`	**setArmSafePosition**`()` Set the arm to a safe position.
`void`	**setBaseStop**`()` Stop any movements of the base.
`void`	**setBaseVelocity**`(double xVel, double yVel, double rotVel)` Set the velocity set-point of the mobile base.
`void`	**setGripperClosed**`()` Close the gripper.
`void`	**setGripperJointPosition**`(double[] values)` Set the position set-point of the gripper joints.
`void`	**setGripperOpen**`()` Open the gripper.

GoalDetector interface method summary

Modifier and Type	Method and Description
`java.lang.String`	**getColor**`()` Get the colour of the object that should be detected.
`com.robb19y.grasping.PointCloudPreProcessing`	**ReadVisionSensor**`()` Returns the actual point cloud perception of the goal object.
`void`	**setColor**`(java.lang.String color)` Set the colour of the object that should be detected.

ObstacleDetector interface method summary

Modifier and Type	Method and Description
`java.util.ArrayList<double[]>`	**getPosition**`()` Returns the position of the actual detected obstacles (expressed in robot mobile frame).

Appendix C

Automatic calibration of the depth camera relative frame position

During the autonomous pick and place tasks, the depth camera is placed with a fixed support above the mobile platform of the KUKA youBot at a certain height z_C with respect to the robot reference frame. Furthermore, the camera is oriented toward the floor with angle β_C to better look at the objects that have to be manipulated (see Figure C.1). In this appendix an automatic way to estimate the values of parameters z_C and β_C, starting from the floor observation, is presented.

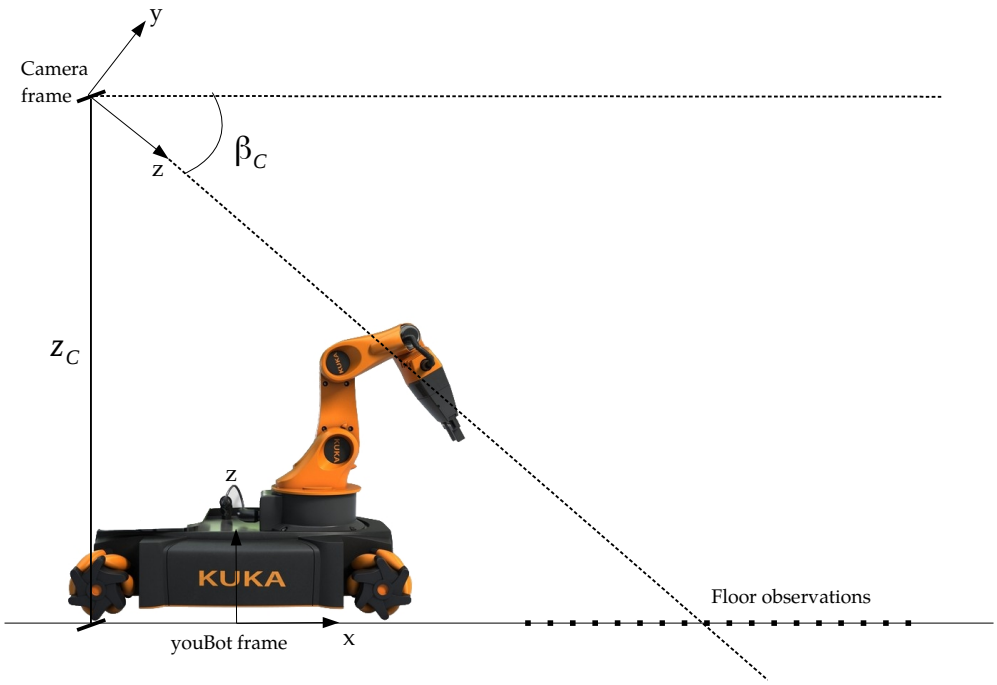


Figure C.1 **Camera reference frame and parameters used for the calibration.**

Given a depth scansion of the floor with N pixel, each having position $p_i = <x_i, y_i, z_i>$ in the reference frame of the camera, linear regression can be used to estimate the straight line that best match the y_i and z_i coordinates of the points. Indeed, the angular coefficient m of that regression line corresponds to the angle β_C, which can be calculated as:

$$\beta_C = atan(m) \quad . \tag{C.1}$$

Using the well known *Least Square* resolution formula [42], the angular coefficient of the regression line can be derived as:

$$m = \frac{\sum_{i=1}^{N} (z_i - \bar{z})(y_i - \bar{y})}{\sum_{i=1}^{N} (z_i - \bar{z})^2} \tag{C.2}$$

with $\bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i$ and $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$.

Parameter z_C, instead, can be calculated considering the average of the floor observations heights, after having aligned them with the floor horizontal line, namely after a rotation of angle β_C :

$$z_C = -\frac{1}{N}\sum_{i=1}^{N} y_i \cos(\beta_C) - z_i \sin(\beta_C) \quad . \tag{C.3}$$

Appendix D

Torque control of KUKA youBot arm

In order to consider the dynamic behaviour of a robotic manipulator, torque control can be used as an effective control strategy, as it allows to set directly the torques and forces exerted by the joint motors. In fact, besides positions and velocities of the joints, it is necessary to control the forces that they exert, so that the whole manipulator structure can have different degrees of stiffness and compliance. Stiffness can be defined as the capacity of a robot to keep rigidly its position even when external forces are applied on it. Compliance, instead, allows the manipulator to change its posture and to deform according with external forces applied on it. In this work, torque control has been adopted to perform manual guidance of the KUKA youBot arm. In this way it is possible to set manually postures of the manipulator, without the effort of programming them. Thanks to the developed torque control strategy, the youBot arm maintains the postures manually assigned, but still it wont fall or collapse due to the gravitational pull.
The parameters, necessary to define the dynamic model of the youBot arm, are reported in the following table.

	Mass [Kg]	**Inertia Tensor Elements [Kg m^2]**			**Centre of mass position [m]**
		I_{XX}	I_{YY}	I_{ZZ}	
Link 1	1.39	0.002952	0.006009	0.0058821	[-0.016, -0.0735, 0]
Link 2	1.318	0.003114	0.0005483	0.003163	[-0.077, 0, 0]
Link 3	0.821	0.0017276	0.0004196	0.0018468	[-0.067, 0, 0]
Link 4	0.769	0.006764	0.0010573	0.000661	[0, 0, 0.062]
Link 5	0.678	0.001934	0.001602	0.00689	[0, 0, -0.062]

Table D.1 **KUKA youBot arm dynamic parameters**

Given the dynamic parameters and the kinematic model of a manipulator, *Euler-Newton* or *Lagrangian* formulation can be used to derive the manipulator dynamic model [61]:

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+\boldsymbol{g}(\boldsymbol{q})=\boldsymbol{\tau} \tag{D.1}$$

where the symbols $\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}$ and $\boldsymbol{\tau}$ denote n-dimensional vectors of joint position, velocity, acceleration and torque variables, respectively, where n is the number of degrees of motion freedom of the robot mechanism. $\boldsymbol{B}(\boldsymbol{q})$ is an $n\times n$ symmetric, positive-definite matrix, and is called the joint-space inertia matrix. $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is an $n\times n$ matrix such that $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is the vector of Coriolis and centrifugal terms (collectively known as velocity product terms); and $\boldsymbol{g}(\boldsymbol{q})$ is the vector of gravity terms.
Many control strategy, as for example *Inverse Dynamics* or *Robust Control* [62], are available to consider the dynamic behaviour of a robotic manipulator inside the robot controller. For the purpose of performing the manual guidance on the youBot arm, it has been sufficient to adopt the simple control law:

$$\boldsymbol{\tau}=\boldsymbol{g}(\boldsymbol{q}) \tag{D.2}$$

which ensures to the robot the capacity to compensate the effects of the gravitational pull and leaves the possibility to manually move all the links of the manipulator in any desired position.

Bibliography

[1] F. Jovane, Y. Koren, C. R. Boër: Present and Future of Flexible Automation: Towards New Paradigms, CIRP ANNALS-MANUFACTURING TECHNOLOGY, vol. 52/2, pp 543-560, 2003.

[2] S. Chitta, E.G. Jones, M. Ciocarlie, K. Hsiao: Mobile Manipulation in Unstructured Environments: Perception, Planning, and Execution, Robotics & Automation Magazine, IEEE (Volume:19 , Issue: 2) , 2012.

[3] S. Bøgh, M. Hvilshøj, M. Kristiansen, O. Madsen: Autonomous Industrial Mobile Manipulation(AIMM): From Research to Industry, Proceedings of the 42nd International Symposium on Robotics, .

[4] S. Bogh, B. Schou, T. Ruehr, Y. Kogan Doemel, Andreas ; Brucker, Manuel, C. Eberst, R. Tornese, C .Sprunk, G. D. Tipaldi, T. Hennessy: Integration and Assessment of Multiple Mobile Manipulators in a Real-World Industrial Production Facility, Proceedings of ISR/Robotik 2014; 41st International Symposium on Robotics, 2014.

[5] H. I. Christensen, P. Case: Mobile Manipulation for Everyday Environments, International Conference on Control, Automation and Systems 2008, 2008.

[6] M. Hvilshøj, S. Bøgh: "Little Helper" - An Autonomous Industrial Mobile Manipulator Concept, International Journal of Advanced Robotic Systems, 2011.

[7] KIVA System: A Different kind of Material Handling Company & Complete Warehouse Automation Solution, http://www.kivasystems.com/about-us-the-kiva-approach/, 2014.

[8] L. C. Wang, L. S. Yong, M. H. Ang: Hybrid of Global Path Planning and Local Navigation implemented on a Mobile Robot in Indoor Environment , Proceedings of the 2002 IEEE International Symposium on Intelligent Control, 2002.

[9] S. Russell, P. Norvig: Artificial Intelligence: A Modern Approach (3rd Edition), Prentice Hall, 2009.

[10] O. Brock, J. Kuffner, J. Xiao: Handbook of Robotics - Chapter 26 Motion for Manipulation Task, Springer, 2008.

[11] R. Bischoff, E. Prassler: KUKA youBot – a mobile manipulator for

research and education, IEEE International Conference on Robotics and Automation (ICRA), 2011.
[12] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo: Robotics - Modelling, Planning and Control, Springer, 2009.
[13] B. Siciliano, O. Khatib: Handbook of robotics, chapter 11 - Kinematically Redundant Robot, Springer-Verlag, 2008.
[14] A. De Luca, G. Oriolo, P. R. Giordano: Kinematic modelling and redundancy resolution for nonholonomic mobile manipulators, IEEE International Conference on Robotics and Automation, 2006.
[15] B. Siciliano, O. Khatib: Handbook of robotics, chapter 1 - Kinematics, Springer, 2008.
[16] S. Sharma,G. K. Kraetzschmar,C. Scheurer,R. Bischoff: Unified Closed Form Inverse Kinematics for the KUKA youBot, ROBOTIK 2012, 2012.
[17] Huatao Zhang, Yunyi Jia and Ning Xi: Sensor-based Redundancy Resolution for a Nonholonomic MobileManipulator, 2012 IEEE/RSJ International Conference onIntelligent Robots and Systems, 2012.
[18] G. Buizza Avanzini, A. M. Zanchettin, P. Rocco: Reactive Constrained Model Predictive Control for Redundant Mobile Manipulators, International Conference on Intelligent Autonomous Systems, IAS 2014, 2014.
[19] B. Siciliano, L. Sciavicco, L. Villani,G. Oriolo: Robotics - Modelling, Planning and Control -3.5.1 Redundant Manipulators, Springer, 2009.
[20] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, R. Dillmann: Manipulability Analysis, IEEE-RAS 12th International Conference on Humanoid Robots, 2012.
[21] J. Nocedal, S. Wright: Numerical optimization - Chapter 3 Line Search Methods, Springer, 1999.
[22] H. Durrant-Whyte, T. Bailey: Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms, Robotics & Automation Magazine, IEEE (Volume:13 , Issue: 2) , 2006.
[23] B. Siciliano, O. Khatib: Handbook of robotics - Chapter 5 Motion Planning, Springer, 2008.
[24] Wikipedia: Dijkstra's algorithm, http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm, 2014.
[25] Wikipedia: A* search algorithm, http://en.wikipedia.org/wiki/A*_search_algorithm, 2014.

[26] O. Khatib: Real-time obstacle avoidance for manipulators and mobile robots, Robotics and Automation. Proceedings. 1985 IEEE International Conference, 1986.
[27] T.Liddy , Tien-Fu Lu , P. Lozo, D. Harvey : Obstacle Avoidance Using Complex Vector Fields, Proceedings of the 2008 Australasian Conference on Robotics and Automation, 2008.
[28] A. T. Miller, P. K. Allen: GraspIt! A versatile simulator for robot grasping, IEEE Robotics & Automation Magazine, 2004.
[29] A. T. Miller, S. Knoopt, H. I. Christensen, P. K. Allent: Automatic GraspPlanning Using Shape Primitives, Proceedings of ICRA, 2003.
[30] Dan Ding, Yun-Hui Liu, Shuguo Wang: Computing 3-D Optimal Form-Closure Grasps, Robotics and Automation, 2000. Proceedings. ICRA '00, 2000.
[31] C. Ferrari, J. Camy: Planning optimal grasps, Robotics and Automation, 1992. Proceedings., 1992.
[32] P. Violero, I. Mazon, M. Tayx: Automatic Planning of a Grasp for a "Pick and Place" Action, 1990 IEEE International Conference on Robotics and Automation, 1990. Proceedings, 1990.
[33] Zhixing Xue, A. Kasper, J. M. Zoellner, R.Dillmann: An Automatic GraspPlanning System for Service Robots, International Conference on Advanced Robotics, 2009. ICAR 2009, 2009.
[34] N. Curtis, Jing Xiao: fficient and Effective Grasping of Novel Objects through Learning and Adapting a Knowledge Base, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008, 2008.
[35] Kimitoshi Yamazaki, Masahiro Tomono, Takashi Tsubouchi, Shin'ichi Yuta: A Grasp Planningfor Picking up an Unknown Object for a Mobile Manipulator, Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, 2006.
[36] C. Goldfeder, P. K. Allen, C. Lackner, R. Pelossof: rasp Planning viaDecomposition Trees, IEEE International Conference on Robotics and Automation, 2007.
[37] D. Fischinger, M. Vincze: Empty the Basket - A Shape Based Learning Approach forGrasping Piles of Unknown Objects, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.
[38] B. Calli, W. Caarls, Q. Lei, M. Wisse, P. Jonker: SMAG: Simultaneous Modeling and Grasping, Robotics Science and Systems (RSS), 2013.

[39] Farrokh Janabi-Sharifi: Automatic Grasp Planning for Visual-Servo Controlled Robotic Manipulators, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 1998.
[40] N. Vahrenkamp: Simultaneous Grasp and Motion Planning, IEEE Robotics & Automation Magazine, 2012.
[41] Point Cloud Library: Removing outliers using a Statistical Outlier Removal filter, http://pointclouds.org/documentation/tutorials/statistical_outlier.php, 2012.
[42] S. M. Ross: Introductory Statistics - Chapter 12 Linear Regression, Academic Press, 2010.
[43] V. D. Nguyen: Constructing stable grasps, IEEE International Conference on Robotics and Automation. Proceedings., 1987.
[44] S. Chitta, E. G. Jones, M. Ciocarlie, K. Hsiao: Mobile Manipulation in Unstructured Environments, IEEE Robotics & Automation Magazine (Volume:19 , Issue: 2), 2012.
[45] Wikipedia: Sequential function chart, Sequential function chart (SFC), 2014.
[46] N. J. Mitra, N. Gelfand, H. Pottmann, L. Guibas: Registration of Point Cloud Data from a Geometric Optimization Perspective, Eurographics Symposium on Geometry Processing, 2004.
[47] A. V. Segal, D. Haehnel, S. Thrun: Generalized-ICP, Robotics: Science and Systems, 2009.
[48] M. Duckham, L. Kulik, M. Worboys, A. Galton: Efficient generation of simple polygons forcharacterizing the shape of a set of points inthe plane, Pattern RecognitionVolume 41, Issue 10, October 2008, Pages 3224–3236, 2008.
[49] Wikipedia: Point in polygon, http://en.wikipedia.org/wiki/Point_in_polygon, 2014.
[50] Mathworks: Simulink, simulation and Model-Based design, www.mathworks.com/products/simulink/, 2014.
[51] Coppelia Robotics: V-Rep, Virtual robot experimental platform, http://www.coppeliarobotics.com/, 2014.
[52] KUKA youBot developers: KUKA youBot API, http://www.youbot-store.com/youbot-developers/software/libraries/kuka-youbot-api, 2014.
[53] Wikipedia: OpenNI, http://en.wikipedia.org/wiki/OpenNI, 2014.

[54] ASUS: Xtion PRO LIVE, http://www.asus.com/Multimedia/Xtion_PRO_LIVE/, 2014.
[55] KUKA youBot developers: ROS Wrapper for KUKA youBot API, http://www.youbot-store.com/youbot-developers/software/frameworks/ros-wrapper-for-kuka-youbot-api, 2014.
[56] Gazebo: Robot simulation made easy, http://gazebosim.org/, 2014.
[57] Benjamin Keiser: Torque Control of a KUKA youBot Arm, Master Thesis at Robotics and Perception Group University of Zurich, 2013.
[58] Roberto Ancona: Navigation and Redundnacy Resolution with KUKA youBot (Simulation)., https://www.youtube.com/watch?v=z0gLybCeIqE, 2014.
[59] Roberto Ancona: Mobile Manipulation in a completely unknown environment with KUKA youBot, https://www.youtube.com/watch?v=j_5ABJsX_Po, 2014.
[60] Roberto Ancona: Mobile Manipulation with KUKA youBot (Simulation environment) , https://www.youtube.com/watch?v=kO8cLKj3e_k, 2014.
[61] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo: Robotics: Modelling, Planning and Control, Chapter 7 - Dynamics, Springer, 2008.
[62] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo: Robotics: Modelling, Planning and Control, Chapter 8 - Motion control, Springer, 2008.